Title:
Advanced Geometry Modeling with Julia Plasm.jl

Authors:
Alberto Paoluzzi, apaoluzzi@os.uniroma3.it, Roma Tre University
Giorgio Scorzelli, scrgiorgio@gmail.com, SCI, University of Utah

Keywords:
Computational Topology, Chain Complex, Solid Modeling, Julia, Arrangement, Boolean Algebras

DOI: 10.14733/cadconfP.2025.235-239

Introduction:
We addressed the challenge of developing a modern functional metalanguage for Solid Modeling in Julia. Engineers created Solid Modeling as a rigorous and universal language for geometry-based engineering. In this context, Plasm (Programming Language for Solid Modeling) was developed several years ago and now stands out as an *engineering metalanguage* implemented in the Julia language, providing a robust framework for defining, representing, and manipulating solid geometries.

The *Julia* Plasm package introduces essential topological methods for solid and geometric modeling computations in Building Information Modeling (BIM) and Computer-Aided Design (CAD). These methods include calculating space partitions, adjacency relations, mesh cell ordering and integration, as well as performing solid set operations using algebraic topology, combinatorial techniques, and Boolean algebra.

Julia Plasm.jl is a package for *advanced geometric programming* specifically designed to generate complex engineering and architectural models represented as assemblies of maps of topological polyhedra. This includes various types of shapes, such as piecewise linear approximations of curved objects and cellular complexes made of simplicial, cubic, and polyhedral cells, typical of the AEC (Architecture, Engineering, and Construction) domain.

The language is the Julia [1] implementation of the original *PLaSM* (Programming Language for Solid Modeling) project [2, 3], funded by the Italian CNR to advance building design and support the momentum of industrialized construction. This project introduced a geometric type for the FL language at the function level, developed by Backus and his team at IBM Almaden in the 1990s. Plasm was initially written in Common Lisp and later ported to Scheme, C++, and Python before being adapted to *Julia*, where the language has found an ideal environment to leverage its mathematical foundation rooted in algebraic topology and Boolean algebra [4, 5].

It isn't easy to compare *Julia* Plasm to existing and past research. We believe it should primarily be compared to the graphics libraries that defined the standardization of the field, such as GKS, PHIGS+, OpenInventor, OpenGL, and DirectX, which have been the more commonly used graphics APIs since the 1990s, along with their recent GPU versions like Metal and Vulkan. However, *Julia* Plasm is a concise geometric API for Julia and may also serve as a functional metalanguage for any geometric application domain, particularly for BIM. Additionally, it is the only language that is strongly oriented toward solid modeling, providing a unique approach based on computational topology and binary Boolean algebra.

This talk introduces, through examples, the creation of geometric models for solid objects using Plasm.jl and the primary primitive operators that Plasm imports from the original FL-based geometric

language into Julia. In Fig. 1, we present three pictures extracted from a small Plasm script (shown in Fig. 2 with comments referencing to the pictures). The Plasm script is the *complete* implementation of a small and simple design project, starting from the lateral design measurements in the directions of x, y, and z. Their Cartesian product produces a cellular complex named idea, comprised of $4 \times 2 \times 2$ 3D cells, whose 1-skeleton is depicted in Fig. 1a. Then, the extracted 2-skeleton is displayed in Fig. 1b. Finally, a well-engineered structural framework featuring reversed beam foundations is shown in Fig. 1c. The Plasm code that produces all wireframe, foil 3D, and solid models is shown in Figure 2.

It is worth noting that this code can be executed as-is as a unique copy-paste action within the REPL (Read-Eval-Print-Loop) interpreter of Julia v1.10 after downloading the open-source package Plasm.jl from GitHub pages at https://github.com/scrgiorgio/Plasm.jl.

Geometric and solid modeling:

Geometric models define the physical appearance of architecture, engineering, and construction (AEC) products at any scale, from structural and envelope components to entire buildings and built environments. They are utilized for design, collaboration, tenders, and contracts. We ported the functional language *Julia* Plasm.jl to Julia to enhance the design, model generation, and visualization of geometric objects of any complexity.

Cellular models, often called *meshes* in engineering, design, and graphics, utilize *discrete cells* to represent a geometric domain. In fields such as geospatial mapping, computer vision, robotics, graphics, finite element analysis, medical imaging, 3D printing, solid modeling, and geometric design, computing incidences, adjacencies, and mesh cell orderings typically depends on various incompatible data structures and algorithms. Conversely, Plasm can be used in all such domains as a *metalanguage* to develop and integrate mutually compatible Julia geometric objects and tools with minimal development effort.

This Julia package introduces the unifying topological concept of *chain complex* to compute 2D or 3D space partitions induced by collections of 1D and 2D geometric objects, addressing these challenges. Such space decompositions, known as *arrangements* in combinatorial topology and algebra, form the foundation of our algebraic approach to geometric and solid modeling.

We simply embed topological relations into Julia's *sparse matrices and vectors* to efficiently manage large geometric datasets. These are employed to compute the chain complex—a collection of linear spaces of cell *chains* and the *linear transformations* between them, spanning dimensions zero to three.

This geometry representation is based on two multidimensional Julia's user geometric data types: *Hpc* (Hierarchical Polyhedral Complex) for generating assemblies and interactive visualization and *Lar* (Linear Algebraic Representation) for storing chain complexes and computing spatial arrangements and Boolean algebras. These features make *Julia* Plasm a versatile and efficient tool for various geometric application domains.

The geometric framework:

Julia Plasm.jl is quick and user-friendly. It utilizes Julia's advanced functional programming features to improve performance and usability. At its core are paradigmatic primitives, generators, and transformers for solid modeling. Furthermore, the package includes homological operators implemented as sparse Julia matrices, crucial for topology and geometry computations.

Parametric models and topological operators demonstrate how to create flexible and reusable designs using parametric functions and topological transformations. Geometric mapping of complexes and grids lets us explore techniques to map and manipulate geometric data structures effectively.

Linearized approximations of curved manifolds are used by Plasm for generation of curves, surfaces, and solids as PL approximations for 1D, 2D, and 3D geometries. Through practical examples and detailed explanations, we strive to equip Julia with the tools and understanding necessary to harness a full potential for advanced geometric modeling.

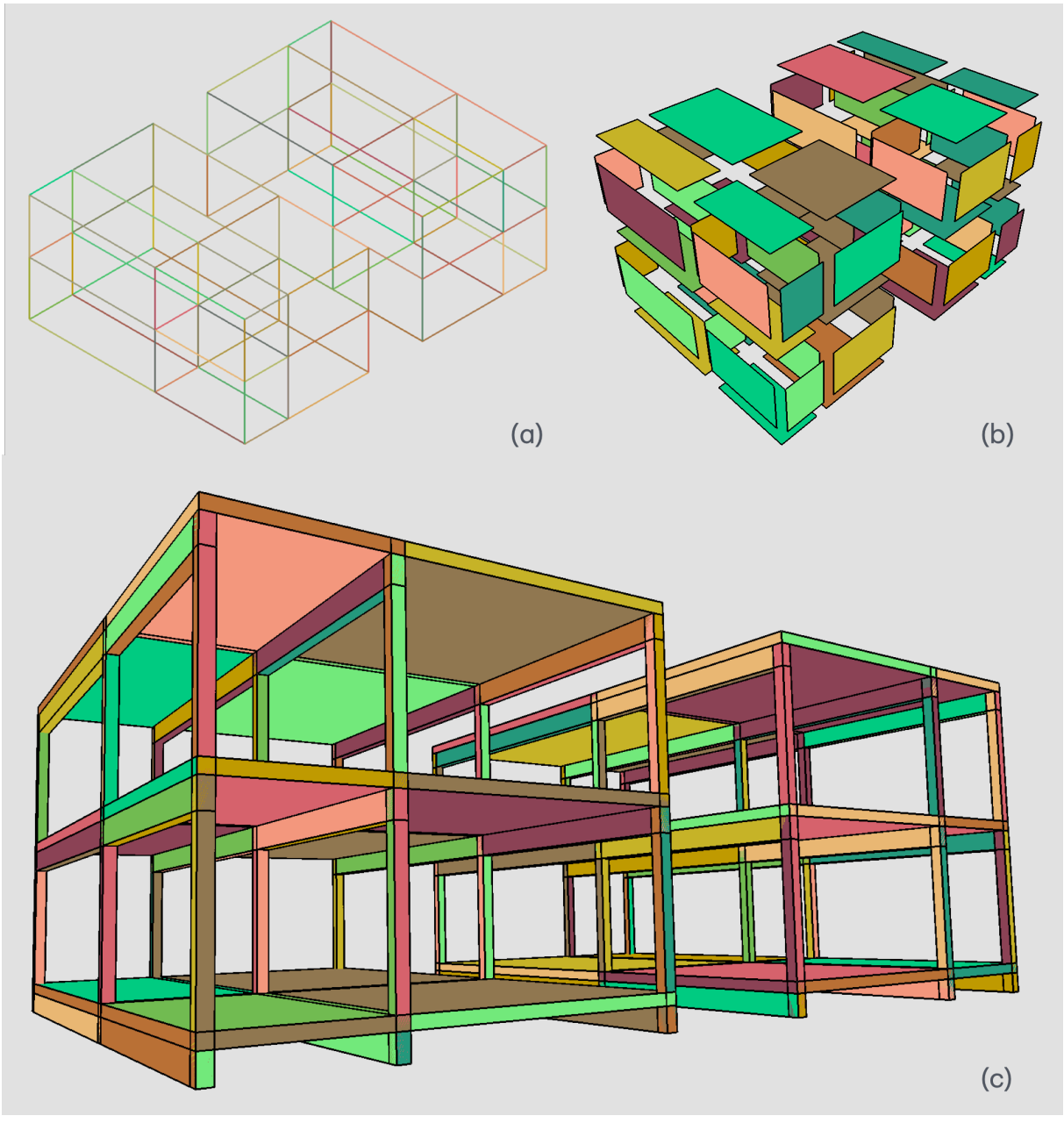

Fig. 1: (a) SK(1)(idea); (b) exploded object 2-complex; (c) structural framexyz model from the project named idea in the following Julia Plasm fragment. All the generating Plasm code is shown in Fig. 2.

```
using Plasm
X = GRID([2.4,4.5,-3,4.5,2.4]);
Y = GRID([7,5]);
Z = GRID([3,3]);
idea = X * Y * Z;
VIEW(SK(1)(idea))   # Fig. 1a
VIEWCOMPLEX(LAR(idea), explode = [1.2,1.2,2.0] )    # Fig. 1b

building110 = X * Y * SK(0)(Z);
building1_101 = SK(1)(SK(0)(X)*SK(1)(Y)*SK(1)(Z));
building1_011 = SK(1)(SK(1)(X)*SK(0)(Y)*SK(1)(Z));

floors = OFFSET([.2,.2,.2])(building110);
framex = OFFSET([.2,.2,.2])(building1_011);
framey = OFFSET([.2,.2,-.4])(building1_101);
framexyz = STRUCT(framex, framey, floors);
VIEWCOMPLEX(LAR(framexyz))                    # Fig. 1c
```

Fig. 2: The Julia Plasm code fragment producing all the images of Fig. 1 .

In particular, *Julia* Plasm features a novel approach to the Boolean algebra of solid models, rooted in the algebraic topology of piecewise-linear geometry. Specifically, it introduces computational topology algorithms to uncover the two- or three-dimensional space partitions (*arrangements*) produced by collections of 1D and 2D geometric objects, respectively.

The package offers straightforward methods for constructing parametric assemblies, where other objects can be treated as actual parameters. Each model is typically generated in its reference frame and then transferred through elementary rotations, scalings, translations, and other means to integrate different component models into the correct assembly aggregation. Specifically, a straightforward syntax allows for *geometric transformations* and *assemblies*.

Plasm.jl also includes functions for calculating *domain integrals* of polynomials on piecewise-linear polyhedra, which is an essential solid modeling tool. In particular, it has simple primitives for determining the mechanical properties of solids, including area, volume, centroid, products, and moments of inertia of models, presented in both scalar and tensor forms.

Boolean algebra from arrangements of space:

Julia Plasm's key topics include the algebraic relationships between solid Boolean algebras with union, intersection, difference, and xor, as well as constructive solid geometry, by using boundary representations of piecewise-linear (PL) cell-decomposed polyhedra. We show how arranging a set of spatially instantiated primitive shapes corresponds to the finite Boolean algebra of regularized sets, encompassing all possible Boolean expressions generated from a given set of constructor primitives.

The more advanced aspect of *Julia* Plasm is the theoretical foundations and practical application of Boolean algebras in modeling solid shapes. We show how the package enables the creation and manipulation of complex geometric models by leveraging homological theory and set algebra. This approach is similar to Constructive Solid Geometry (CSG), expanded with n-ary operations and explicit representation of external space.

In this talk we aim to give a quick understanding of the mathematical principles and computational techniques underlying *Julia* Plasm's solid modeling capabilities. In particular, we demonstrate that resolving any solid algebra expression with a finite number of geometric primitives, once the spatial arrangement and certain point memberships are executed, algebraically reduces to Julia's set operations

on bit strings, sparse matrix-vector multiplication, and shape reconstruction from algebraic topological atoms.

Alberto Paoluzzi, https://orcid.org/0000-0002-3958-8089
Giorgio Scorzelli, https://orcid.org/0000-0002-6263-2738

References:

[1] J. Bezanson, A. Edelman, S. Karpinski, V. B. Shah. "A Fresh Approach to Numerical Computing," *SIAM Review*, vol. 59, no. 1, pp. 65–98, Jan 2017. [Online]. Available: http://doi.org/10.1137/141000671.

[2] A. Paoluzzi, V. Pascucci, and M. Vicentino, "Geometric programming: a programming approach to geometric design," *ACM Trans. Graph.*, vol. 14, no. 3, pp. 266–306, Jul. 1995. [Online]. Available: http://doi.acm.org/10.1145/212332.212349

[3] A. Paoluzzi, *Geometric Programming for Computer Aided Design*. Chichester, UK: John Wiley Sons, 2003. [Online]. Available: https://doi.org/10.1002/0470013885

[4] A. Paoluzzi and G. Scorzelli, "Computational topology, boolean algebras, and solid modeling," *Computer-Aided Design*, vol. 181, p. 103839, 2025. [Online]. Available: https://doi.org/10.1016/j.cad.2025.103839

[5] ——, *BIM geometry with Julia Plasm: Functional language for CAD programming*. Springer Nature, June 2025, ISBN 978-3-031-90243-7. [In print]. Available: https://link.springer.com/book/9783031902437