EUROGRAPHICS
CONFERENCE 2024

TOPOLOGICAL DATA STRUCTURE FOR COMPUTER GRAPHICS

Gábor Fábián
ELTE Eötvös Loránd University, Budapest, Hungary

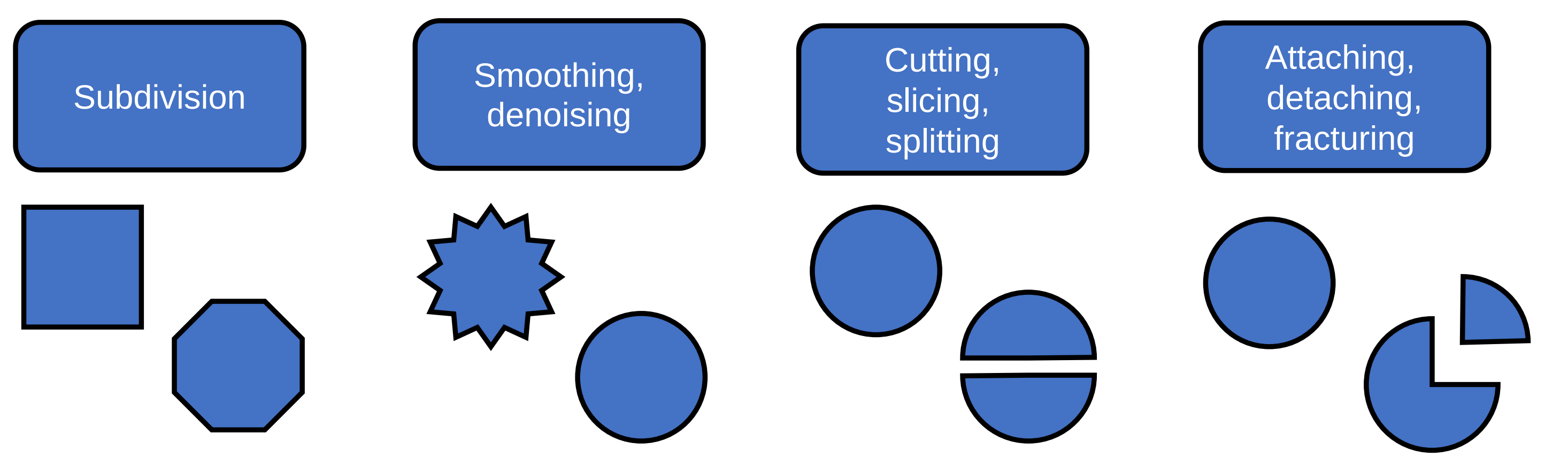
PROBLEM

This research is motivated by the following well-known contradiction. In computer-aided design or modeling tasks, we generally represent surfaces using edge-based data structures as winged edge [1], half-edge [3, 6], or quad-edge [4]. In contrast, real-time computer graphics represents surfaces with face-vertex meshes, since for surface rendering, there is no need for the explicit representation of edges.

In most cases, when mainly local modifications are used (e.g. vertex split, edge flip, face removal), traditional winged edge and half-edge data structures perform well. However, for global operations (affecting lots of vertices, edges, faces), the advantages of edge-based data structures seem to diminish. In this research we introduce a novel data structure for representation of triangle meshes. Our representation is based on the concept of face-vertex meshes with adjacencies, but we use some extra information and new ideas that greatly simplify the implementation of algorithms.

GLOBAL OPERATIONS

By global operations, we mean operations that extend to a large part of the mesh, so the number of triangles/vertices traversed due to the operation is comparable to the number of triangles/vertices in the entire mesh. Some examples of global operations are the followings.



The environment that can be destroyed, objects that can be cut are good examples for use-cases when we often need to perform global operations.

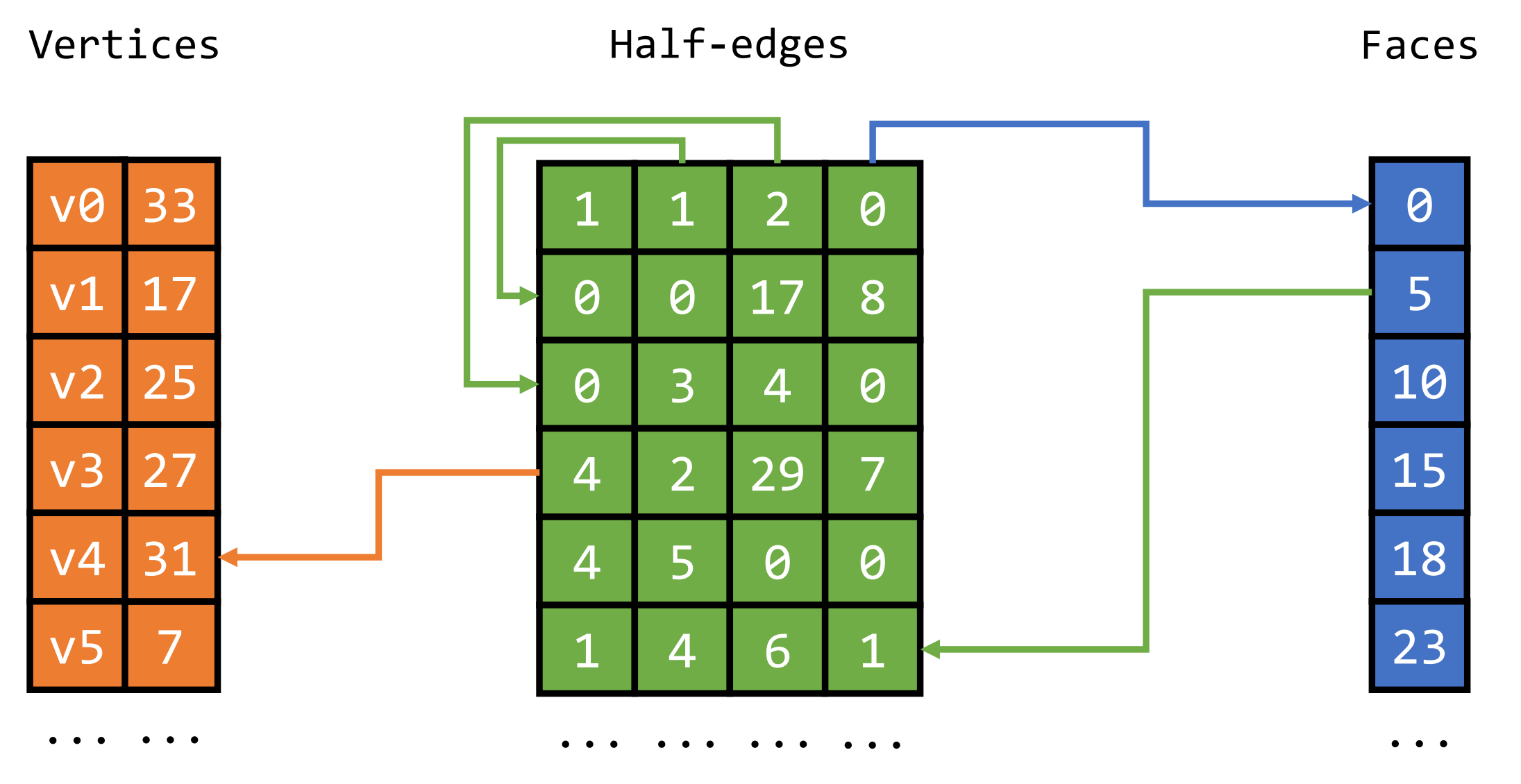
When designing the data structure, we stated the following requirements.

- Explicit use of vertex array & index array
- No edges
- Fixed amount of data stored per vertex / triangle
- Fast global operations

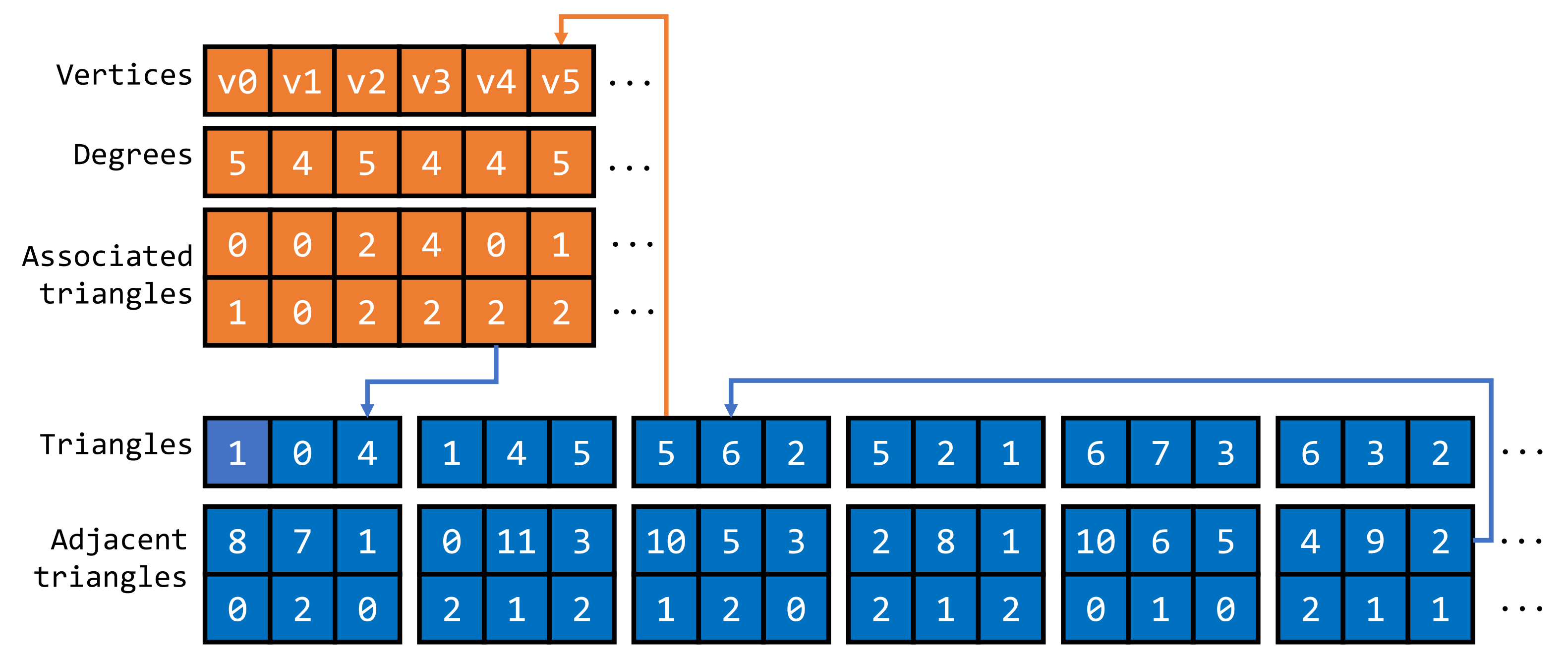
We compared our representation with the industry standard half-edge. We refer to our data structure as SolidMesh, emphasizing that it is suitable only for storing triangulation of surfaces of solid geometries.

COMPARISON OF DATA STRUCTURES

The half-edge data structure allows efficient execution of local modifications, geometric information of a neighborhood of an edge is encapsulated into half-edges. Half-edges store references for their start vertex, the opposite and the next half-edge, and the face with which the half-edge associated. Vertices and faces store references one of their half-edges.



In SolidMesh data structure a vertex stores its degree, and a reference for itself in an arbitrary triangle. Edges are only implicitly represented, any two circularly consecutive vertex of a triangle define an edge. Using this convention, a face stores references to its edge-adjacent triangles and endpoints of its own edges.



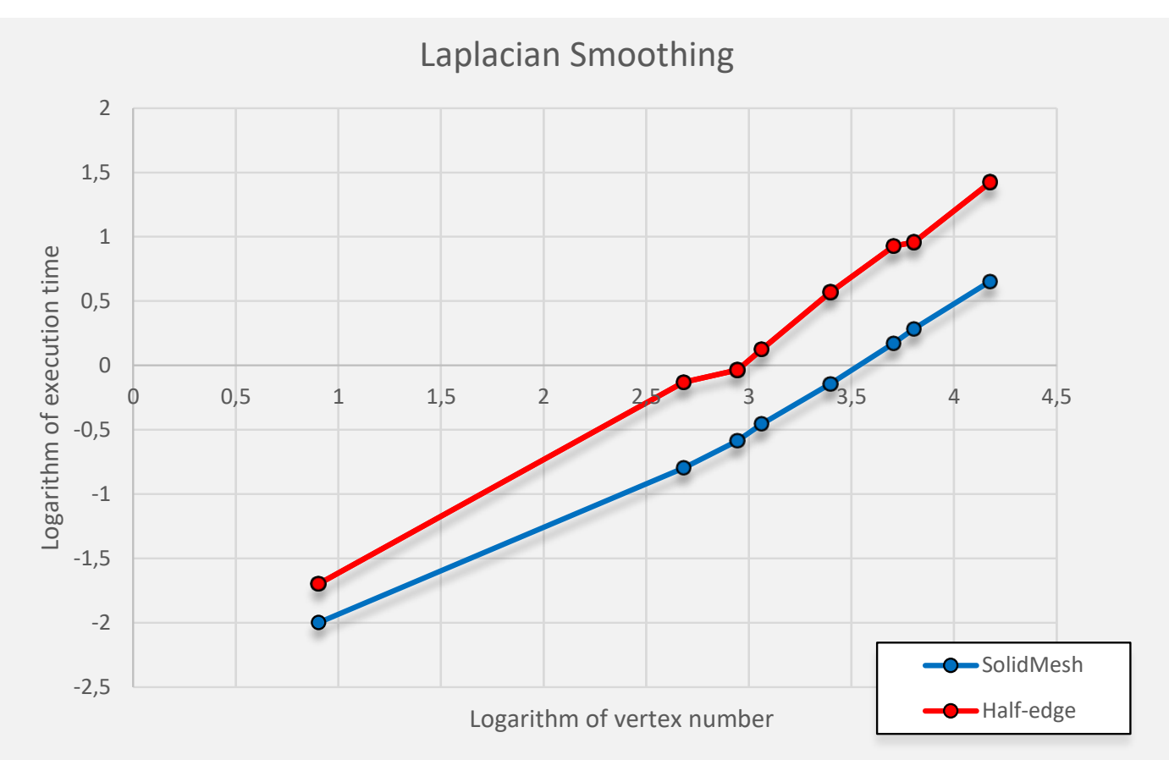
In the implementation of our data structure, we did not create a new face class, which would achieve similar encapsulation. Instead, we added some extra (one- and multi-dimensional) arrays containing all the necessary geometric information to the vertex and index arrays, similarly to the render dynamic meshes [8].

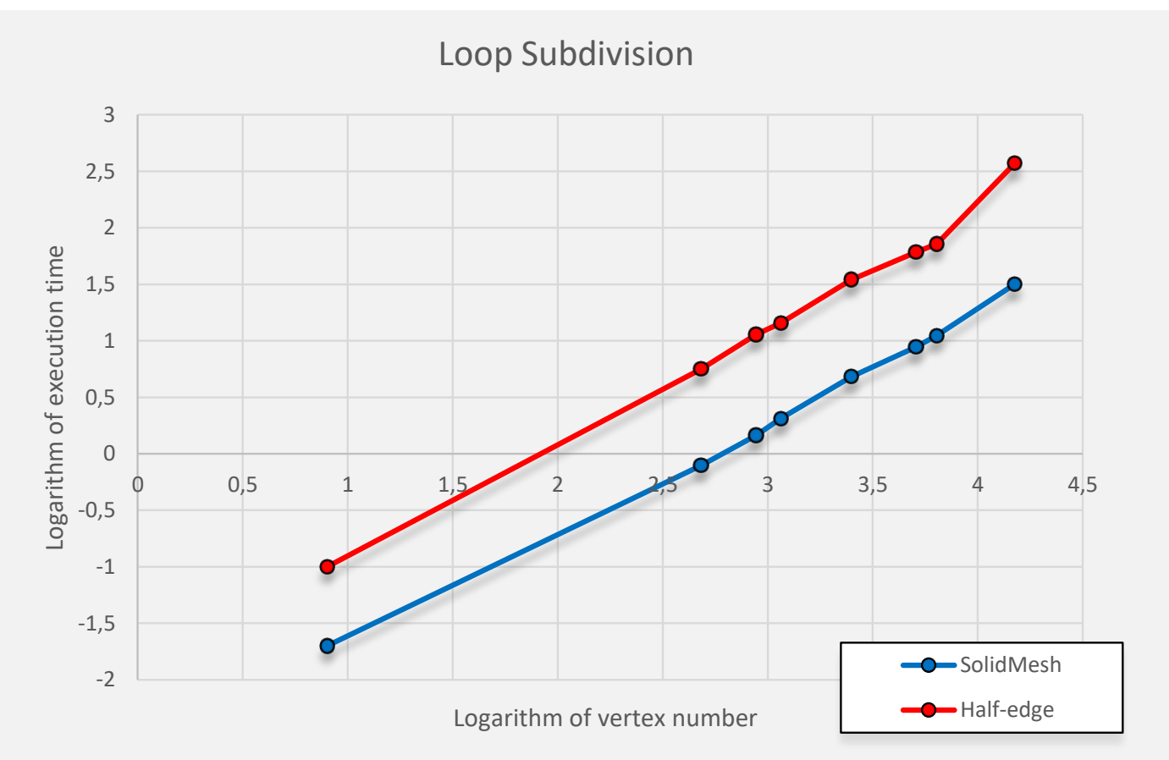
Notice, that SolidMesh data structure explicitly contains the vertex array and the index array that we pass to the GPU for rendering. In addition, it stores all the topological information that sufficient for performing complex mesh operations.

RESULTS

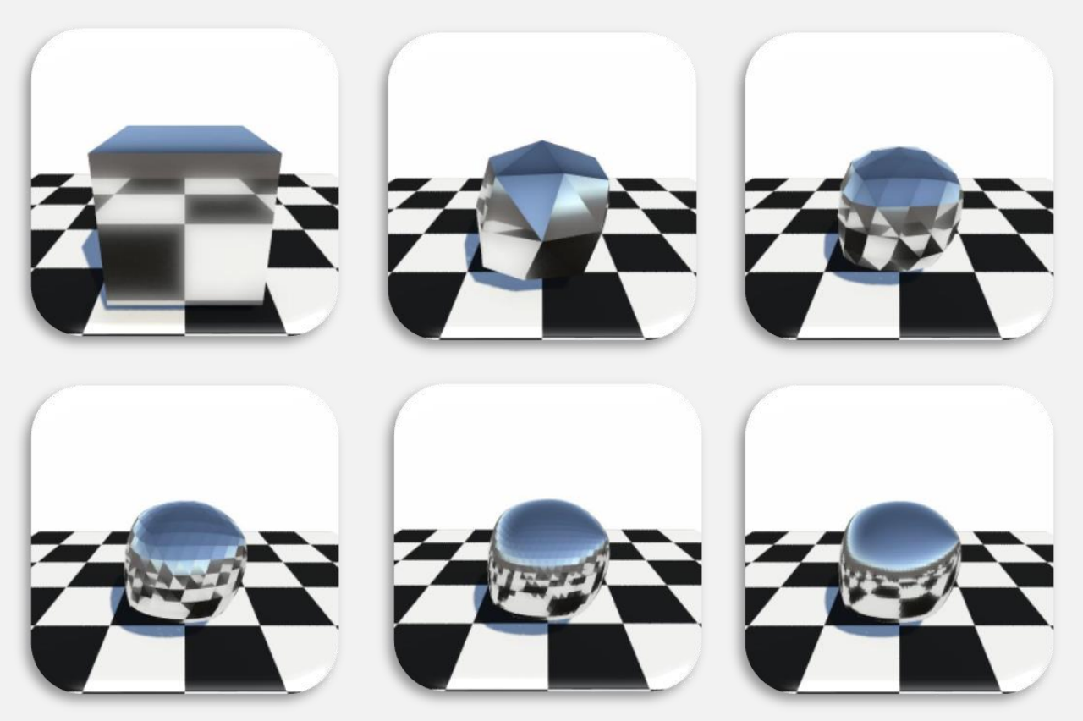
In our experiments we used a half-edge data structure [7], and we implemented our proposed data structure in C# language using Unity game engine [9]. We have done several tests to measure time cost of some local and global operations. We compared the performance of the data structures for some complex operations as smoothing and subdivision. Our results confirmed, that many local and global operations can be easily implemented and efficiently performed without explicit representation of edges. Moreover, our surface representation stores less data than the half-edge data structure.

We give two representative examples, the application of the Laplacian smoothing filter [2] and the Loop subdivision scheme [5]. Laplacian filtering is a global operation, where we iteratively reposition each vertex to the center of gravity of its neighbours. Loop subdivision is another global operation working on polyhedrons defined by triangular faces. The subdivision operation is not trivial; we need to break down each face of the polyhedron and compute the coordinates of each vertex (new and old). By calculation of vertex positions the edge-adjacent triangle pairs play an important role, therefore the half-edge data structure is often chosen for implementing this subdivision scheme.





Model	Vertices	Smoothing time [ms]		Subdivision time [ms]	
		SolidMesh	Half-edge	SolidMesh	Half-edge
Unit Cube	8	**0,01**	0,02	**0,02**	0,10
Sphere	482	**0,16**	0,74	**0,79**	5,63
Cube	602	**0,18**	0,70	**1,01**	7,33
Torusknot	880	**0,26**	0,92	**1,46**	11,36
2-tori	1156	**0,35**	1,33	**2,05**	14,37
Bunny	2503	**0,72**	3,70	**4,81**	34,78
Mug	5084	**1,49**	8,48	**8,83**	60,87
Ducky	6390	**1,93**	9,05	**11,12**	71,99
Armadillo	15002	**4,50**	26,50	**31,74**	370,69



Our results seem to support that, despite the lack of explicit edge representation in our data structure, complex operations can be executed much faster with it. According to our measurements, the Laplacian smoothing and Loop subdivision implemented in SolidMesh data structure ran approximately 10 times faster than the half-edge implementation.

AFFILIATIONS

Új Nemzeti Kiválóság Program

Ministry of Culture and Innovation

ELTE | Faculty of Informatics

Supported by the ÚNKP-23-4 New National Excellence Program of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation Fund.GO #4

REFERENCES

[1] Baumgart, B. G.: A polyhedron representation for computer vision. In Proceedings of the May 19-22, 1975, National Computer Conference and Exposition (AFIPS '75). Association for Computing Machinery, New York, USA, 1975, 589 – 596.
[2] Belyaev, A.; Ohtake, Y.: A comparison of mesh smoothing methods, Israel-Korea Bi-National Conference on Geometric Modeling and Computer Graphics, Tel Aviv University, 83 – 87, 2003.
[3] Campagna, S.; Kobbelt, L.; Seidel H.-P.: Directed edges – A scalable representation for triangle meshes. Journal of Graphics, GPU & Game Tools, 3, 1998, 1 – 11.
[4] Guibas, L.; Stolfi, J.: Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams, ACM Transactions on Graphics, 4(2), 1985, 74 – 123.
[5] Loop, C. T.: Smooth Subdivision Surfaces based on Triangles, M.S. Mathematics thesis, University of Utah, USA, 1987.
[6] Müller, D. E.; Preparata, F. P.: Finding the intersection of two convex polyhedra, Theoretical Computer Science, 7(2), 1978, 217 – 236.
[7] Piker, D.; Pearson, W.: Plankton, GitHub, https://github.com/meshmash/Plankton, accessed 1 March 2024.
[8] Tobler, R. F.; Maierhofer, S.: A Mesh Data Structure for Rendering and Subdivision. In Proceedings of the January 30-February 3, 2006, Winter School of Computer Graphics (WSCG'2006), Plzen, Czech Republic, 2006.
[9] Unity Real-Time Development Platform, https://unity.com/, accessed 1 March 2024.