MESH MORPHING

Jindřich PARUS, Master Degree Programme (5)
Dept. of Computer Science And Engineering, FAV, ZČU
E-mail: jparus@students.zcu.cz

Supervised by: Dr. Ivana Kolingerová

ABSTRACT

A morphing or metamorphosis is the process of continuously transforming one geometric object into another. This technique is used mainly in animation (special effects) and design. Our work aims at 3D geometric objects given in boundary representation. The goal of our work is to develop a method which works more or less automatically (i.e. there is no need to manually specify correspondence between the source and target objects) and which produces realistic animations.

1 METHOD OVERVIEW

We are following ideas proposed in [1] and [2]. The task of morphing is usually formulated as follows. Given two source models M_1 and M_2, the goal is to compute a topology and geometry of the model M(t), where $t \in <0; 1>$, so that the shape of M(0) is the same as the shape of M_1 and the shape of M(1) is the same as the shape of M_2. The model M is a combination of both source models so it can represent shape of the source model and the shape of target model at the same time. In-between phases of morphing are then produced by interpolating extremal vertices positions (i.e. one extreme position is M in the shape of M_1 and second is M in the shape of M_2). Model M is in further text called supermesh. The process of morphing is usually divided into three steps.

1. Finding a correspondence between the source and target model. For purposes of morphing we find a correspondence of vertices, it means which vertex from model M_1 corresponds to which vertex from model M_2.
2. Construction of a supermesh. The supermesh is a model that represents both the source and target object.
3. Interpolation of corresponding vertices. Selecting convenient method of interpolation and finding trajectories for the vertices is a big problem by itself and it is beyond the scope of our work. We consider only a simple linear interpolation.

In the following text we will describe steps 1 (section 2) and 2 (section 3) in more detail.

2 FINDING THE CORRESPONDENCE

This step involves finding a mapping of the vertices from model M_1 to the surface of model M_2 and vice versa. This situation is shown in Fig.1.

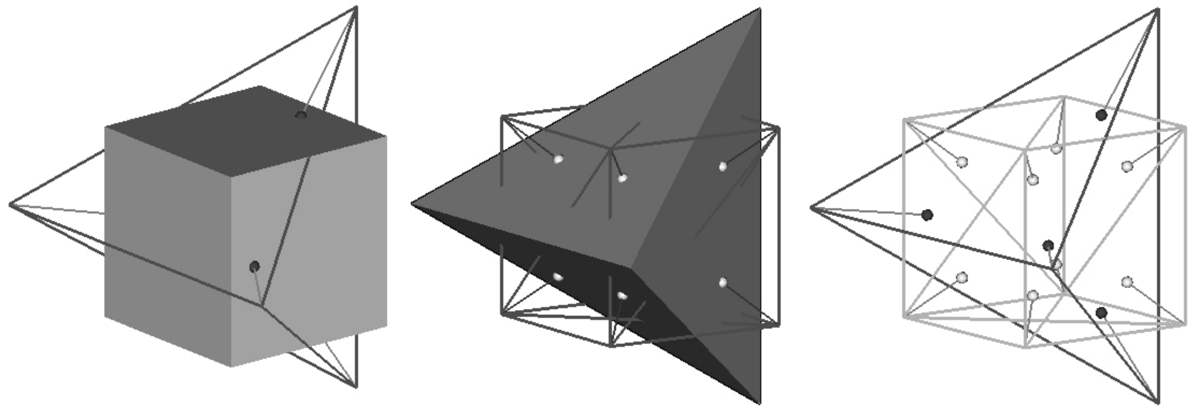

Fig. 1: *Mapping of the vertices from M_1 to the surface of model M_2 and vice versa.*

As we can see, in a general case a vertex of model M_1 is mapped somewhere on the surface of model M_2, more concretely on some triangle. The question is how to find this particular triangle. In the area of morphing this is usually solved by mapping the source objects into a common parameter domain. The natural parameter domain for genus 0 meshes (i.e., without a hole) is a unit sphere. For star-shaped polyhedra we can use a spherical projection. If we want to handle a wider class of objects we have to find a mapping that maps bijectively source models to the parameter domain (i.e., spring embedding [1] or harmonical maps [3]).

If we are dealing only with star-shaped polyhedra, we can use the spherical projection from a star-point. The star-point lies inside a kernel of the polyhedron and it can be found by intersecting of halfspaces (halfspaces are bounded by model faces).

After the parametrization of the models, we can find the position of a particular vertex of the model M_1 on the surface of the model M_2. For point location, we can use standard techniques, but we have to adapt them for spherical triangles because the triangles of the model became spherical triangles in projection. For representing a position of a vertex relatively to the triangle we use barycentric coordinates.

3 SUPERMESH CONSTRUCTION

Once we have found a mapping in the previous step, we can start to construct the supermesh M. As we have said before, the supermesh is a combination of both source and target object; combination in that sense that the M has topology of M_1 and M_2. This will be in the further text called „shared topology“.

In this step we will merge both spherical projections. This is also often called „a map overlay computation“. In Fig.2 we can see how to compute a shared topology.

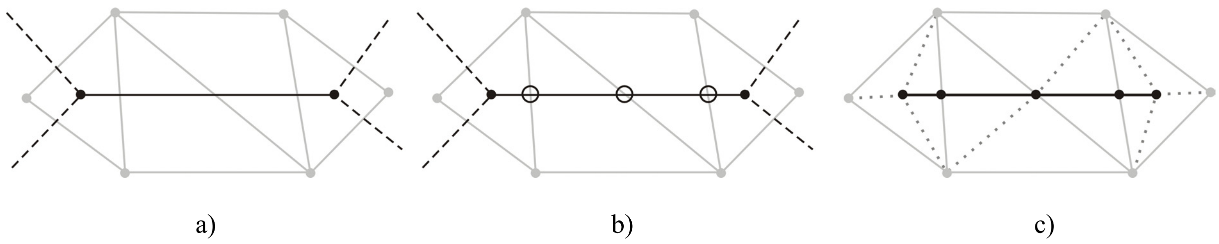

Fig. 2: *Shared topology: a) Parts of two source models. b) Intersections. c) Resulting shared topology with the newly inserted edge (bold), dotted edges are the edges added in order to have the resulting shared topology triangulated.*

Fig. 2a) shows parts of two models, for a simplicity the M_2 model is represented by one edge only. To achieve the shared topology, we have to insert edges of the model M_2 into the topology of the model M_1. In Fig. 2b) we can see that we have to find intersections of the edges of the model M_2 with the edges of the model M_1. Once we have found all intersections, we can insert the edge of the model M_2 into the model M_1. Newly inserted edge is marked bold in Fig.2c). In Fig. 2c) we can also see that some more edges have been added in order to have a resulting shared topology triangulated (the process of triangulation will be described in section 3.2).

Now we will explain how to compute intersections between the edges of M_1 and M_2. In a general case the edges of M_1 and M_2 are nonparallel and nonintersecting and they do not intersect, but by projecting the edges into the common parameter domain the edges can intersect. The problem is that the edges became great arcs in the projection. Thus this step involves computing intersection of great arcs. Using a brute force algorithm, we can intersect all edges from M_1 with all edges from M_2, but with $O(N^2)$ complexity. But we can use some kind of “walking” algorithm that has lower expected complexity (one is proposed in [2]).

3.1 MERGING THE SOURCE AND TARGET OBJECT

Till now we have worked with models in their parameterization in the common parameter domain. Also the intersections were computed in the projection. We can project the intersections back to the model edges. As we mentioned before, edges of the source models are in general case nonintersecting and nonparallel, so one intersection in parameter domain produces one vertex on the edge of M_1 and one vertex on the edge of M_2. This situation is depicted below in Fig.3 where we can see how the intersections are projected back to the source and target model.

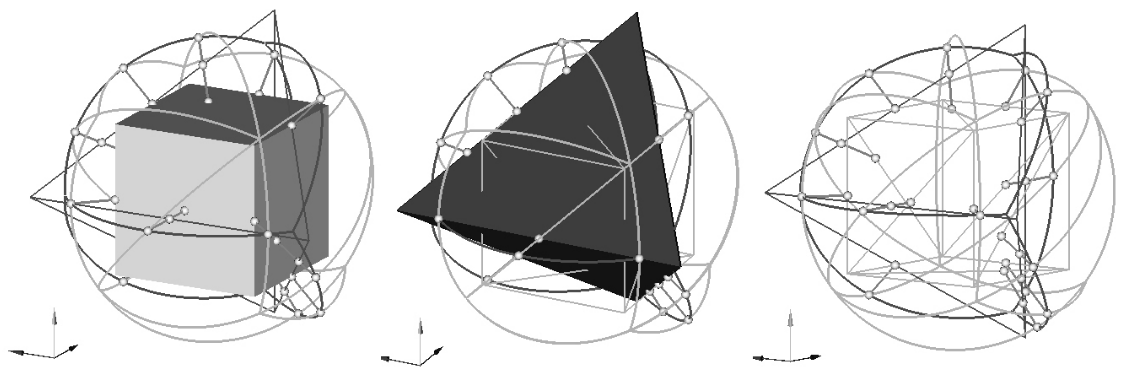

Fig. 3: *Projection of intersections back to the source and target model.*

After finding all intersections, we can merge the objects in these intersections. We can start with merging the sets of edges and vertices of the M_1 and M_2. We do not merge the sets of faces because the resulting set of faces will be different. Information about faces will be extracted after the merging process (this step is described in the section 3.2).

Next, the process of merging goes as follows. First the edges of the source object are splitted in these intersections, this splitting produces new vertices and edges. Both edges and vertices are added to the data structures of the supermesh. Then the target object is processed in a similar way, it means that all edges are also splitted, new edges are added to the supermesh, however, the newly obtained vertices are already present in the supermesh from processing of the source object.

Now we have a supermesh that is a combination of both the source and target models. We have achieved shared topology but the shape of the supermesh is something between M_1 and M_2 as we can see in Fig. 4a).

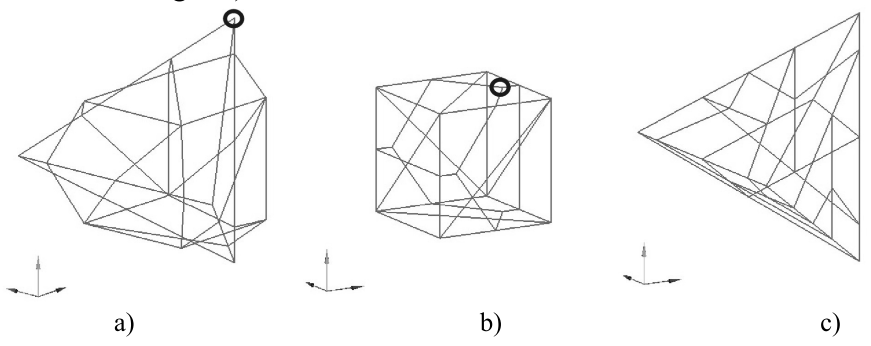

Fig. 4: *The resulting supermesh. a) Non-transformed supermesh. b) The supermesh transformed to the shape of M_1. c) The supermesh transformed to the shape of M_2.*

In Fig.4a) it is possible to see that some vertices do not have the right position with the respect to the shape of M_1 yet. More concretely, badly positioned vertices are former vertices of the model M_2. These vertices have to be transformed to lie on the surface of the model M_1. For this purpose we can use already computed barycentric coordinates of these vertices. Fig. 4b) shows the supermesh transformed into the shape of M_1. The marked vertex from Fig. 4a) is already pushed back to the surface of M_1. Finally Fig. 4c) shows the supermesh transformed into the shape of M_2.

3.2 SUPERMESH TRIANGULATION

In this phase we have the model described by edges. For purposes of shading and other higher-level manipulation, we have to describe the object as a set of triangles. The problem is that the edges form polygons with up to six edges and it is necessary to triangulate them. We cannot use an arbitrary triangulation because the polygons are in general case non-planar. Our method is a modification of the method proposed in [3]. The triangulation method is as follows. For each vertex we maintain a list of incident edges and we have to establish their order, so we sort those edges by angle. Again, edges are not in one plane, so it is necessary to project them to a plane and perform angular sorting in the plane. Then we process the sorted edge fan and if two adjacent edges in the edge list are not connected with an edge, we add a new edge.

In the following section we will present some results of our method.

4 EXAMPLES

Fig. 5 shows various morphing transitions. It can be seen how new faces appear during the morphing process.

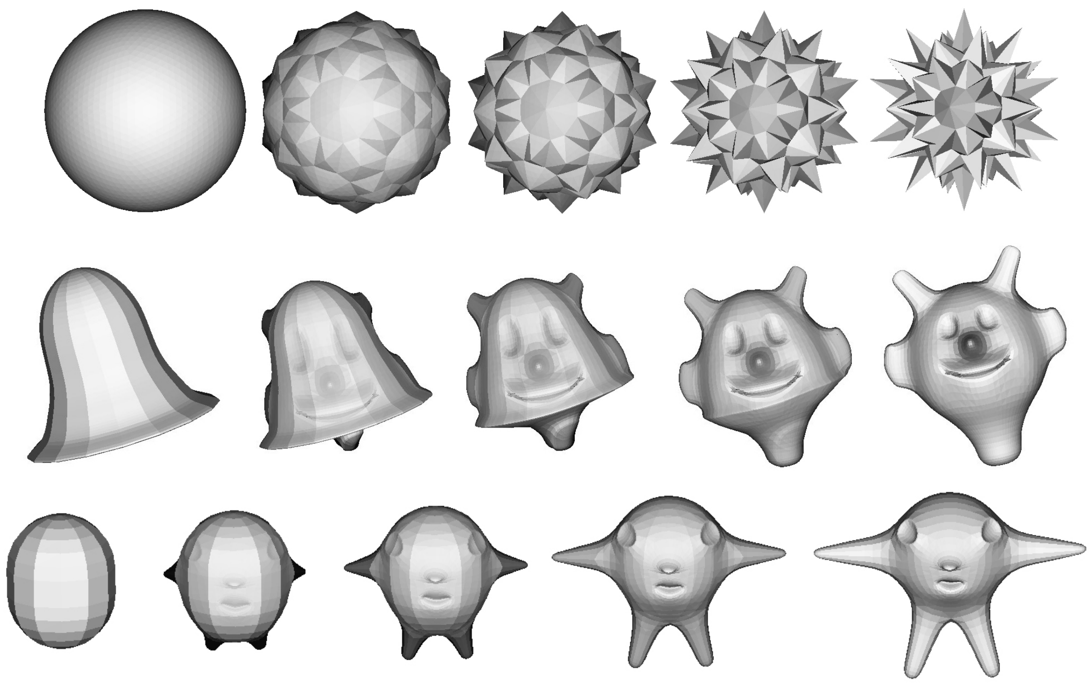

Fig. 5: *Examples of morphing transitions.*

5 CONCLUSIONS

We have presented a method for transforming one object into another so that the transformation looks realistic and do not produce disturbing effects. Now we are able to process only star-shaped objects because of the mapping method and we have to manually specify the star-point. In the future work we would like to handle a wider class of objects. Also we want to allow the user to specify some important corresponding features and exploit this information as much as possible.

ACKNOWLEDGEMENT

This work was supported from the Ministry of Education of The Czech Republic, project MSM 235 200 005. The author would like to thank to Doc. Dr. Ing. I. Kolingerová for professional assistance and to my colleague Ing. P. Vaněček for language correction.

REFERENCES

[1] Alexa, M.: Merging Polyhedral Shapes with Scatered Features, The Visual Computer, Vol. 16, No. 1, pp. 26-37, 2000

[2] Kent, J.R., Carslon W.E., Parents R.E.: Shape Transformation for Polyhedral Objects, Computer Graphics (SIGGRAPH'92) 26, pp. 47-54

[3] Kanai, T., Suzuki, H., Kimura F.: 3D Geometric Metamorphosis based on Harmonic Map, The Visual Computer, Vol. 14, No. 1, pp. 166-176, 1998