SIGRAD (2002)
Mark Ollila (Editors)

Surface Construction with Near Least Square Acceleration based on Vertex Normals on Triangular Meshes

Tony Barrera

Cycore AB
Dragarbrunnsgatan 35, P.O. Box 1401, S-751 44 Uppsala, Sweden

Anders Hast

Creative Media Lab
University of Gävle, Kungsbäcksvägen 47, S-801 76 Gävle, Sweden. aht@hig.se

Ewert Bengtsson

Centre for Image Analysis
Uppsala University, Lägerhyddsvägen 17. S-752 37, Uppsala, Sweden. ewert@cb.uu.se

Abstract

Shading makes faceted objects appear smooth. However, the contour will still appear non smooth. Subdivision schemes can handle this problem by introducing new polygons in the mesh. The disadvantage is that a more complex mesh takes more time to render than a simple one. We propose a new method for constructing a curvilinear mesh using quadratic curves with near least square acceleration. This mesh could be used for subsequent subdivision of the surface. This can be done on the fly, at least in software rendering, depending on the curvature of the contour. The advantage is that new polygons are only inserted where needed. However, in this paper we will focus on how such curvilinear mesh can be constructed using vertex points and vertex normals for each polygon. Thus, information about neighboring polygons are not needed and on the fly subdivision is made easier.

1. Introduction

Recursive surface subdivision has gained a lot of attention in recent years since it makes it possible to have several levels of detail[11]. As an example, the famous Utah teapot was modeled by using Beziér patches in a rather coarse mesh. This mesh could be subdivided into smaller patches creating a tighter mesh, and thus a smoother looking teapot. The new mesh points are created by computing new points on each Beziér surface patch. Nevertheless, many models do not have an underlying curve description that could be used for subdivision. Such faceted models only have a flat polygon mesh. The interior of the object can appear smooth if a shading algorithm is used. Nonetheless, the contour will still appear non smooth since the edges are straight lines. However, it is possible to create a spline surface interpolating or approximating the polygonal mesh, which then can be used for recursive subdivision. This will yield a tighter mesh, which will make the object appear more smooth on the contour.

Several such approaches have been made and the algorithm by Catmull and Clark[2] use information about neighboring polygons. Others only use the vertices and normals at these vertices, when constructing the new surface. Volino and Magnenat-Thalmann[10] construct a spherical surface called a Spherigon. Vlachos et al.[9] use a three sided cubic Beziér surface for their Curved PN triangles and Bruijn[1] uses quadratic Beziér surfaces. Overveld and Wyvill[7] also use Beziér surfaces, however they replace the cubic curve with two quadratic curves in order to guarantee monotonic curvature. In surface subdivision presented by Maillot and Stam[5], a spline surface is generated and the curvature and smoothness is controlled by a parameter α, which is set to some proper value for optimal smoothness. Other subdivision schemes have been introduced by Loop[4] and kobbelt[3].

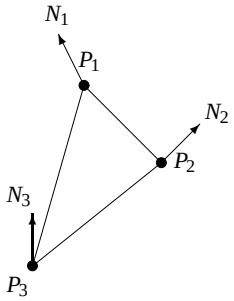

Figure 1: *Three sided polygon with three vertices and three normals*

1.1. Main Contribution

We propose a new algorithm for finding a near least square acceleration curvilinear mesh. And we will focus on three sided polygons for simplicity. This mesh is constructed by using quadratic curves which are derived using vertex normals and vertex points only. Thus, we do not propose a new subdivision scheme. We only propose how the new vertex points can be found by using the curvilinear mesh. Moreover, it will not produce surfaces that are C^1 continuous over polygon edges. This would have required the use of cubic surfaces. However, this method will be faster since it uses quadratic meshes. Hence, the method will yield a surface which is smoother than the original mesh.

2. Second Degree Best-Fit Least Square Acceleration Boundary Curves

The method of finding a least square acceleration boundary curve has been explored before by Overveld and Wyvill[7]. They use cubic Beziér curves that are divided into two quadratic curves. We will show how this can be done for quadratic curves directly, even though the curve will be an approximation of the a least square acceleration curve.

We assume that we have a triangular mesh with coordinate points P_1, P_2, P_3 and corresponding normalized normals $\mathbf{N}_1, \mathbf{N}_2, \mathbf{N}_3$ as shown in figure 1. The tangents corresponding to a pair of normals as shown in figure 2 can be obtained by using the so-called Gram-Schmidt orthogonalization algorithm[6]:

$$\mathbf{T}_1 = \mathbf{N}_2 - \mathbf{N}_1(\mathbf{N}_1 \cdot \mathbf{N}_2), \tag{1}$$

$$\mathbf{T}_2 = -\mathbf{N}_1 + \mathbf{N}_2(\mathbf{N}_1 \cdot \mathbf{N}_2). \tag{2}$$

Note. that the normals are assumed to be normalized. We shall show later that we do not need to normalize the tangent vectors. However, at this point we shall assume that they have unit length. It should also be pointed out that when the angle θ between the normals, as shown in figure 2, is zero, then we can not compute the tangent in this way. The simple solution is to use linear interpoltaion between the surface

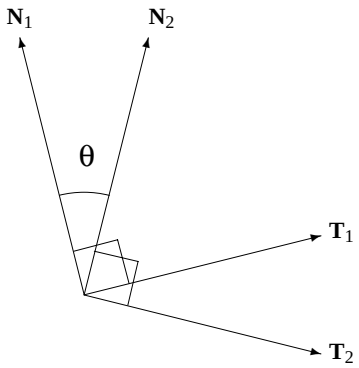

Figure 2: *Construction of the tangent vectors* $\mathbf{T}_1$ *and* $\mathbf{T}_2$ *orthogonal to* $\mathbf{N}_1$ *and* $\mathbf{N}_2$ *respectively, lying in the plane spanned by* $\mathbf{N}_1$ *and* $\mathbf{N}_2$

points instead of a quadratic curve, whenever the angle is zero or very close to zero.

We shall derive a curve spanned by the tangent vectors and vertex points which is as relaxed as possible. This curve can be obtained by minimizing the integral which is the total sum of the square acceleration:

$$\int_0^1 \| f''(t) \|^2 dt \tag{3}$$

on some variable β that controls the tangent length and apply it on the boundary curve. This defines the least square acceleration. However, a second degree curve can only have one derivative determined when it is also determined to pass through two end points. Therefore, we have to find a second degree curve spanned by the tangent vectors and vertex points which have optimal fit between both these tangent vectors and the derivatives. We shall derive such a curve by finding an optimal fit for the first tangent, then we can minimize the integral in equation (3).

2.1. Optimal Fit for the First Tangent

The quadratic curve is given by the equation:

$$\mathbf{f} = \mathbf{a}t^2 + \mathbf{b}t + \mathbf{c}. \tag{4}$$

We have the initial conditions:

$$\mathbf{f}(0) = P_1, \tag{5}$$

$$\mathbf{f}(1) = P_2, \tag{6}$$

$$\mathbf{f}'(0) = \alpha\beta\mathbf{T}_1, \tag{7}$$

$$\mathbf{f}'(1) = \beta\mathbf{T}_2, \tag{8}$$

where $\mathbf{T}_1$ and $\mathbf{T}_2$ is the tangent vectors corresponding to $\mathbf{N}_1$ and $\mathbf{N}_2$ respectively. The constants α and β are to be determined. The value of α will determine optimal (in the least square-sense) fit between the curve and the second tangent. The value of β will determine the least square acceleration of the curve in the interval [0,1]. It can be shown that:

$$\mathbf{c} = P_1, \tag{9}$$

$$\mathbf{a}+\mathbf{b}+\mathbf{c} = P_2, \tag{10}$$

$$\mathbf{b} = \alpha\beta\mathbf{T}_1. \tag{11}$$

Let $\mathbf{P}$ be the vector from P_1 to P_2:

$$\mathbf{P} = P_2 - P_1. \tag{12}$$

Then we have:

$$\mathbf{a} = \mathbf{P} - \alpha\beta\mathbf{T}_1. \tag{13}$$

At the first edge, we have the initial condition that the tangent is equal to the derivative, as shown in equation (7). Furthermore, we want to determine α so that the difference between the derivative and the tangent is as small as possible. Thus:

$$\frac{\partial}{\partial\alpha}\left\{\|\mathbf{f}'(1) - \beta\mathbf{T}_2\|^2\right\} = 0. \tag{14}$$

In this way we seek the least-square difference between the tangent and the derivative. The difference of the curve slope and the polygon tangents in one of the ends equals zero, but at the other end minimization can be performed:

$$\frac{\partial}{\partial\alpha}\left\{(2\mathbf{a}+\mathbf{b}-\beta\mathbf{T}_2)^2\right\} = 0 \Rightarrow \tag{15}$$

$$2\alpha\beta^2\mathbf{T}_1\cdot\mathbf{T}_1 - 4\beta\mathbf{P}\cdot\mathbf{T}_1 + 2\beta^2\mathbf{T}_1\cdot\mathbf{T}_2 = 0. \tag{16}$$

Finally, divide both sides of the equation (16) by -2β and rearrange the terms:

$$2\mathbf{P}\cdot\mathbf{T}_1 - \alpha\beta\mathbf{T}_1\cdot\mathbf{T}_1 - \beta\mathbf{T}_1\cdot\mathbf{T}_2 = 0. \tag{17}$$

In order to find an α that fulfills the criterion set in equation (17) we must also determine β.

2.2. Finding the Least Square Acceleration

Next, we determine β such that the integral of the acceleration is least-square minimum in the interval [0,1]:

$$\frac{\partial}{\partial\beta}\int_0^1 \|\mathbf{f}''(t)\|^2 dt = 0 \tag{18}$$

The second derivative of $\mathbf{f}(t)$ is constant: $\mathbf{f}'' = 2\mathbf{a}$. Since it does not depend on the variable t we have:

$$\frac{\partial}{\partial\beta}\|2a\|^2 = 0 \Rightarrow \tag{19}$$

$$8\alpha^2\beta\mathbf{T}_1\cdot\mathbf{T}_1 - 8\alpha\mathbf{P}\cdot\mathbf{T}_1 = 0. \tag{20}$$

Divide both sides of the equation (20) by -8β and rearrange the terms:

$$\mathbf{P}\cdot\mathbf{T}_1 - \alpha\beta\mathbf{T}_1\cdot\mathbf{T}_1 = 0. \tag{21}$$

This gives us a system with the two equations 17 and 21. The solution becomes:

$$\alpha = \frac{\mathbf{T}_1\cdot\mathbf{T}_2}{\mathbf{T}_1\cdot\mathbf{T}_1}, \tag{22}$$

$$\beta = \frac{\mathbf{P}\cdot\mathbf{T}_1}{\mathbf{T}_1\cdot\mathbf{T}_2}. \tag{23}$$

This is valid for the quadratic curve

$$\mathbf{f}_1(t) = (\mathbf{P} - \alpha\beta\mathbf{T}_1)t^2 + \alpha\beta\mathbf{T}_1 t + P_1. \tag{24}$$

2.3. Near least Square Acceleration

So far we have a curve where the derivative of $\mathbf{f}_1(t)$ at P_1 is the same as the tangent $\mathbf{T}_1$ and the derivative at P_2 is as close to $\mathbf{T}_2$ as the least square minimization allows. If we had chosen to have the same derivative of the function as the tangent at P_2 and optimize at P_1 instead, we would get another curve. Which one will be the best? It is hard to tell. However, we could get an approximation by taking the average of both. Hence, we will get an near least square acceleration second degree curve which is close to optimal in both ends.

2.4. Same Procedure for the Other Side

The next step is to repeat the same procedure for the other tangent $\mathbf{T}_2$ at the other end of the curve, in order to obtain $\mathbf{f}_2(t)$. We use the initial conditions:

$$\mathbf{f}(0) = P_1, \tag{25}$$

$$\mathbf{f}(1) = P_2, \tag{26}$$

$$\mathbf{f}'(0) = \beta'\mathbf{T}_1, \tag{27}$$

$$\mathbf{f}'(1) = \alpha'\beta'\mathbf{T}_2, \tag{28}$$

Then we derive the boundary curve using these conditions, where:

$$\mathbf{c} = P_1, \tag{29}$$

$$\mathbf{a}+\mathbf{b}+\mathbf{c} = P_2, \tag{30}$$

$$2\mathbf{a}+\mathbf{b} = \alpha'\beta'\mathbf{T}_2. \tag{31}$$

This system of equations gives:

$$\mathbf{a}+\mathbf{b} = P_2 - P_1 = \mathbf{P}, \tag{32}$$

$$\mathbf{a} = \alpha'\beta'\mathbf{T}_2 - \mathbf{P}, \tag{33}$$

$$\mathbf{b} = 2\mathbf{P} - \alpha'\beta'\mathbf{T}_2. \tag{34}$$

Finally, the quadratic curve becomes:

$$\mathbf{f}_2(t) = (\alpha'\beta'\mathbf{T}_2 - \mathbf{P})t^2 + (2\mathbf{P} - \alpha'\beta'\mathbf{T}_2)t + P_1. \tag{35}$$

Repeating the minimization process of $\mathbf{f}_2(t)$ for both β and α gives:

$$\alpha' = \frac{\mathbf{T}_1 \cdot \mathbf{T}_2}{\mathbf{T}_2 \cdot \mathbf{T}_2}, \tag{36}$$

$$\beta' = \frac{\mathbf{P} \cdot \mathbf{T}_2}{\mathbf{T}_1 \cdot \mathbf{T}_2}. \tag{37}$$

2.5. Putting it all together

By taking the mean value of (24) and (35) we get a symmetric curve with near least square acceleration:

$$\mathbf{f}_3(t) = \left(\frac{\alpha'\beta'\mathbf{T}_2 - \alpha\beta\mathbf{T}_1}{2}\right)t^2 + \left(\mathbf{P} + \frac{\alpha\beta\mathbf{T}_1 - \alpha'\beta'\mathbf{T}_2}{2}\right)t + P_1. \tag{38}$$

It is easy to prove that it will be symmetric. The derivative is:

$$\mathbf{f}_3'(t) = (\alpha'\beta'\mathbf{T}_2 - \alpha\beta\mathbf{T}_1)t + \left(\mathbf{P} + \frac{\alpha\beta\mathbf{T}_1 - \alpha'\beta'\mathbf{T}_2}{2}\right). \tag{39}$$

Moreover, the derivatives at the edges are:

$$\mathbf{f}_3'(0) = \mathbf{P} + \frac{\alpha\beta\mathbf{T}_1 - \alpha'\beta'\mathbf{T}_2}{2}, \tag{40}$$

$$\mathbf{f}_3'(1) = \mathbf{P} - \frac{\alpha\beta\mathbf{T}_1 + \alpha'\beta'\mathbf{T}_2}{2}. \tag{41}$$

Clearly, the derivatives will be symmetric around $\mathbf{P}$ which is the vector between the vertex points. Since the curve is quadratic we can not get a curve that has derivatives equal to the tangents at the vertex points. However, for a quadratic curve, this is as close as we can get with the least square acceleration requirement.

2.6. Invariance with respect to Normalization

We stated earlier that the tangent vectors do not need to be normalized and we shall prove that this is true. Expand the terms $\alpha\beta\mathbf{T}_1$ and $\alpha'\beta'\mathbf{T}_2$:

$$\alpha\beta\mathbf{T}_1 = \frac{\mathbf{T}_1 \cdot \mathbf{T}_2}{\mathbf{T}_1 \cdot \mathbf{T}_1}\frac{\mathbf{P} \cdot \mathbf{T}_1}{\mathbf{T}_1 \cdot \mathbf{T}_2}\mathbf{T}_1 = \frac{\mathbf{P} \cdot \mathbf{T}_1}{\mathbf{T}_1 \cdot \mathbf{T}_1}\mathbf{T}_1, \tag{42}$$

$$\alpha'\beta'\mathbf{T}_2 = \frac{\mathbf{T}_1 \cdot \mathbf{T}_2}{\mathbf{T}_2 \cdot \mathbf{T}_2}\frac{\mathbf{P} \cdot \mathbf{T}_2}{\mathbf{T}_1 \cdot \mathbf{T}_2}\mathbf{T}_2 = \frac{\mathbf{P} \cdot \mathbf{T}_2}{\mathbf{T}_2 \cdot \mathbf{T}_2}\mathbf{T}_2. \tag{43}$$

Remember that $\mathbf{P}$ is an vector and not a point and therefore equation (42) is the projection of $\mathbf{P}$ on $\mathbf{T}_1$. Likewise, equation (43) is the projection of $\mathbf{P}$ on $\mathbf{T}_2$. Hence, normalization of $\mathbf{T}_1$ and $\mathbf{T}_2$ is not necessary, since projection of one vector onto another is independent of the other vectors length.

3. Quadratic curvilinear surfaces

A quadratic surface can be determined by using the six control points[8] in figure 3. The edge mid points $\mathbf{P}_{12}, \mathbf{P}_{23}$ and $\mathbf{P}_{13}$ can be determined by setting $t = 0.5$ in (38). By using a quadratic surface, it is possible to use any kind of subdivision scheme without recalculating new points in a recursive manner. Instead, new points are retrieved from the quadratic surface.

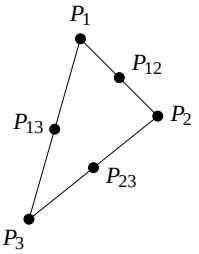

Figure 3: *Six control points*

4. Rendering the Curvilinear Polygons

Figure 4 shows a wire frame rendering of a Torus. It is quite obvious that the Torus has got straight edges on the outer contour. These can be made smooth by using the proposed method. Figure 5 shows the same Torus, but this time, the curvilinear mesh is used. The contour is now smooth.

To our knowledge, most modern hardware do not have the capability to insert new polygons on the fly. The only possible solution, in order to make the contour smooth, is to subdivide the whole object before rendering. However, the result is a more complex object. Moreover, it will take more time to render it since it has more polygons. It would be preferable if we could subdivide the object on the fly, only where extra polygons are needed, i.e on the contour. The proposed method could be used to produce these extra polygons that will make the object smoother.

The extra points will lie on the quadratic surface, that will have near least square acceleration. This will assure that the contour will be as relaxed as possible using quadratic boundary curves. However, since quadratic surfaces are used, C^1 continuity is not guaranteed. If the angle between the normals is large, then the constructed curve can make a rather large bend. This implies that there is a sharp edge between the polygons or that the polygons should have been subdivided prior rendering. This is of course a problem common also for shading, not only for this method.

However, subdivision on the fly is not as easy as one might think at first glance. First of all, we must be able to determine which polygons lie on the contour. This can be done if pointers to neighboring polygons are available. Then, it is possible to check if a neighboring polygon is back facing. Secondly, it is possible that a polygon edge that lies behind another polygon can have a large curvature and if subdivision is performed, then it will be visible. As long as it is

considered being hidden, it is no problem but when it eventually lies on the contour, due to that the object is rotating, it will suddenly pop up as a heavily curved contour. This problem is generally known as popping.

We suggest these problems for further research. Nonetheless, the proposed method could be used for determining new points for any subdivision scheme that will subdivide an object in order to make it appear more smooth.

Vlachos et al.[9] suggest that their PN triangles should be subdivided on the graphics chip without any software assistance. The proposed method could also be used for hardware subdivision since it is based on the available vertex information for each polygon. Furthermore, it should be an attractive method for hardware implementation, since the bottleneck in todays graphics hardware is in feeding the graphics processor with triangles at a sufficient rate. Thus, fewer triangles could be transfered and more triangles are created on the fly on the graphics chip. This can not easily be done for the contour only, since information from neighboring polygons are needed. Instead, on the fly subdivision is done for the whole object.

Another interesting fact is that the dot product of $\mathbf{P}$ and the tangents $\mathbf{T}_1$ and $\mathbf{T}_2$ is invariant under rotation and translation. Hence, these products could be precomputed. This is probably better for software rendering where it is possible to save computation time. For a hardware implementation, this means that two new variables must be transferred through the graphics pipeline and this will make the bottleneck problem previously discusses even worse.

5. Conclusions

It has been shown that a curvilinear mesh with near least square acceleration for quadratic surfaces could be constructed from vertex normals and vertex points. This mesh can then be subdivided using an appropriate subdivision algorithm. The proposed method will be fast, since the computed tangents does not need to be normalized. Furthermore, it has been shown that some terms can be precomputed in order to speed up software rendering.

5.1. Future Work

We propose for future work, how the presented method for constructing a curvilinear mesh, could be used in order to make on the fly subdivision for contours only. Moreover, it should be easy to use the method for on the fly subdivision on the graphics chip. This would decrease the problem of the bandwidth bottleneck.

It should also be determined if this method is good enough for more complex objects, or if cubic surfaces are preferred, even though they are computationally more expensive. Furthermore, it should be investigated whether the average function (38) could be replaced with either of the functions (24)

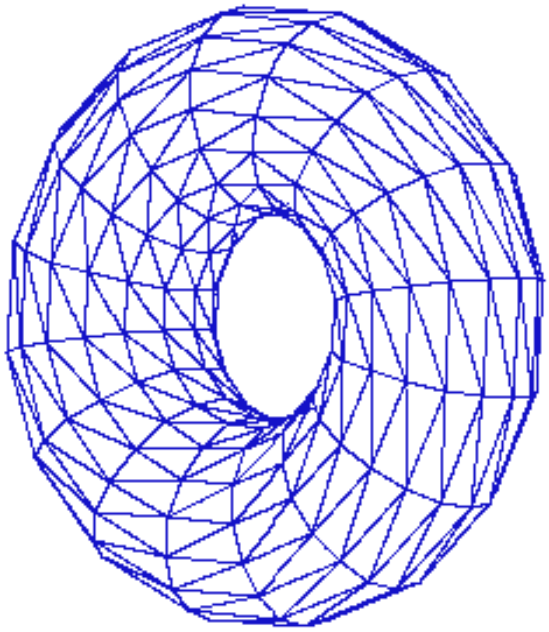

Figure 4: *A wire frame rendering of a Torus*

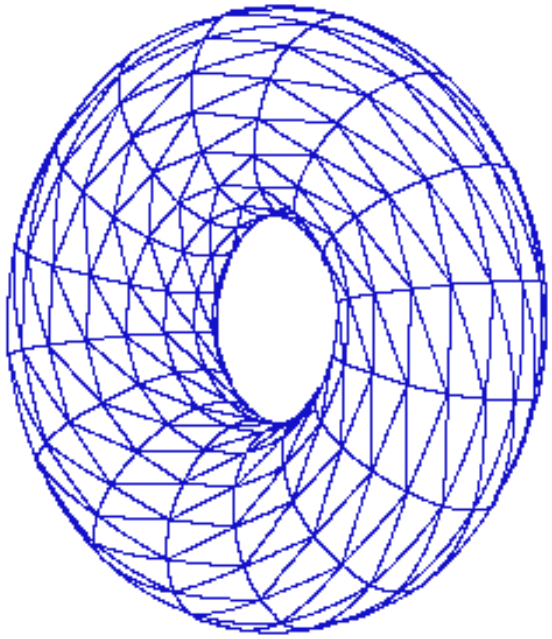

Figure 5: *A curvilinear mesh for a Torus*

and (35) depending on how the normals are pointing compared to the edge.

References

1. J. Bruijn. Quadratic Bezier triangles as drawing primitives *Proceedings of the 1998 EUROGRAPHICS/SIGGRAPH workshop on Graphics hardware 1998 , Lisbon, Portugal* pp. 15 - 24, 1998
2. E. Catmull, J. Clark. Recursively generated B-spline Surfaces on arbitrary Topological Meshes. *Computer Aided Design*, 10 pp.350 - 355, Oct 1978.
3. Leif Kobbelt. $\sqrt{3}$-subdivision *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 103 - 112, July 2000.
4. Charles Loop. Smooth subdivision surfaces based on triangles. *Master s thesis, University of Utah, Department of Mathematics*, 1987.
5. J. Maillot, J. Stam. A unified Subdivision Scheme for Polygonal Modeling *Proceedings Eurographics 2001*, Vol 20, No 3, pp. 471-479, 2001.
6. W. K. Nicholson. Linear Algebra With Applications *PWS Publishing Company*, Third Edition, pp. 275, 1995.
7. K. van Overveld and B. Wyvill. An Algorithm for Polygon Subdivision Based on Vertex Normals *Proceedings Computer Graphics International 1997*, pp. 3-12, 1997.
8. L. Seiler. Quadratic Interpolation for Near-Phong Quality Shading *Proceedings of the conference on SIGGRAPH 98: conference abstracts and applications*, Page 268, 1998.
9. A. Vlachos , J. Peters , C. Boyd , J. L. Mitchell. Curved PN triangles *Proceedings on 2001 Symposium on Interactive 3D graphics, March 2001* pp. 159-166, 2001.
10. P. Volino, N. Magnenat-Thalmann. The SPHERIGON: A Simple Polygon Patch for Smoothing Quickly your Polygonal Meshes. *Computer Animation'98, proceedings, IEEE Press*, pp.72-79, 1998
11. A. Watt. 3D Computer Graphics. *Addison-Wesley*, pp.59-64, 2000.