SPLINES AND SUBDIVISION

MODELING AND ANIMATION

Dan Englesson

Wednesday 18th May, 2011
19:02

Abstract

In this lab which was done in the course TNM079, Modeling and animation in spring 2011 at Linkpings university, curve and mesh subdivision was implemented as well as a localized evaluation of the analytical spline curve. The subdivision curve and analytical curve was compared and shown that the subdivision curve converge to the analytical curve in only a few iterations. The loop subdivision scheme is a nice and understandable subdivision algorithm which is straight forward to implement and yields very pleasing results. Adaptive subdivision was also implemented and the mean magnitude of the three mean curvature estimate from the three vertices spanning up the face was used in order to determine which faces should be subdivided or not. It yielded good results where it subdivided more at faces where the mesh was changing more and no subdivision at the faces that had a lower value than a certain threshold. This saves a lot of extra triangles where subdivision is not needed.

1 Introduction

The lab was done in the course TNM079, Modeling and animation, in spring 2011 and was performed to get a better understanding of different representations of curves and also subdivision of both curves and meshes. Uniform subdivision of a spline curve was implemented and by iteratively subdivide and visually compared with the analytically spline curve, which was also improved to only have a localized evaluation of the Bspline curve. The Loop subdivision scheme was implemented in this lab as well as a adaptive subdivision scheme based on mean curvature. Both curves and subdivision are very important tools in the game- and movie-industry and many others.

2 Method

2.1 Implement curve subdivision

In this task uniform cubic spline curve subdivision had to be implemented and be visually compared with the analytically evaluated spline curve [1].

$$S = \frac{1}{8}\begin{pmatrix} 8 & 0 & 0 & 0 & 0 \\ 4 & 4 & 0 & 0 & 0 \\ 1 & 6 & 1 & 0 & 0 \\ 0 & 4 & 4 & 0 & 0 \\ 0 & 1 & 6 & 1 & 0 \\ 0 & 0 & 4 & 4 & 0 \\ 0 & 0 & 1 & 6 & 1 \\ 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 0 & 8 \end{pmatrix} \tag{1}$$

The full cubic subdivision matrix S in equation (1) can be reduced to two rules instead, see equation (2), except for the boundary points which are taken care of in an other way.

$$\begin{aligned} c'_i &= \tfrac{1}{8}\left(1c_{i-1} + 6c_i + 1c_{i+1}\right) \\ c'_{i+\frac{1}{2}} &= \tfrac{1}{8}\left(4c_i + 4c_{i+1}\right) \end{aligned} \tag{2}$$

The c'_i is the re-weighted coefficient from the old ones, marked black in figure 1, and $c'_{i+\frac{1}{2}}$ is the newly created coefficients that are added in between the old coefficients during the subdivision, marked gray in figure 1. This means that the number of points generated at each subdivision step is as in equation (3), where N is the number of points.

$$N_{new} = N + N - 1; \tag{3}$$

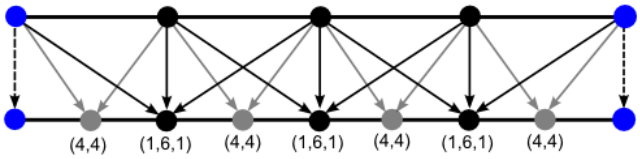

Figure 1: A re-weighting scheme for cubic spline subdivision, blue dots are the boundary points, black the re-weighted old points and the gray points is the newly added points. The weights from the old points are given in the parenthesizes.

The boundary coefficients are evaluated as in equation (4), which are the coefficients ignored by equation (2). The boundary condition in equation (4), states that the endpoints will not be re-weighted or changed during subdivision.

$$\begin{aligned} c'_0 &= c_0 \\ c'_{end} &= c_{end} \end{aligned} \tag{4}$$

Equation (2) and equation (4) were implemented in the function Subdivide() in UniformCubicSplineSubdivisionCurve.cpp where equation (2) is performed in one while-loop looping over all coefficients except for the boundary points, begin and end, which are taken care of by the boundary condition equation (4).

2.2 Implement mesh subdivision

In this assignment as stated an implementation of mesh subdivision was performed, where the mesh subdivision was done according to the Loop subdivision scheme, which is C^2 continuous at regular vertices and G^1 at least else where.

The Loop subdivision is performed like this. For each triangle/ face four new triangles are created within the face, see figure 2.

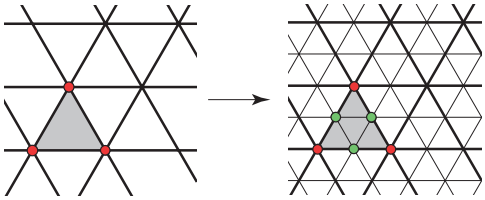

Figure 2: The triangle refinement rules for Loop subdivision.

The newly created ”edge” vertices, seen in green in figure 2, are placed by the weighted average of the four ”old” vertices, neighboring this vertex. The neighboring vertices spanning up the edge where the ”edge” vertex is placed are weighted by a factor of $3/8$ and the two vertices further away are weighted by a weight of $1/8$,see figure 3 on the left for visual clarification.

The newly created vertices, ”edge” vertices, where created in the LoopSubdivisionMesh.cpp in the function EdgeRule, where caution had to be taken during calculation of these vertices and weights so less round-off error would occur. Equation (5) demonstrates how it was implemented in the EdgeRule to reduce the round-off errors, where v_0 and v_1 are the vertices closest to the new ”edge” vertex. If the weights and vertices are placed correctly there wont be an error in the mesh, creating holes. However if one places the weights and the vertices together as one normally would do, errors will appear in the mesh in forms of holes. These errors are caused by very small round-off errors and will treat a single vertex as two if the error is to large and therefore create holes.

$$\boldsymbol{v}_{new} = 0.375(\boldsymbol{v}_0 + \boldsymbol{v}_1) + 0.125(\boldsymbol{v}_2 + \boldsymbol{v}_3) \tag{5}$$

$$\boldsymbol{v}_{new} = (1 - k\beta)\boldsymbol{v} \sum_{i=0}^{k} \beta \boldsymbol{v}_{ng}(i) \tag{6}$$

The new positions for the ”old” points, seen in red in figure 3, are calculated with equation (6), where the k is the valance, which means the number of neighboring vertices, and $\boldsymbol{v}_{ng}$ is an array of the k neighboring vertices. For a visual comparison of equation (6), see figure 3 to the right.

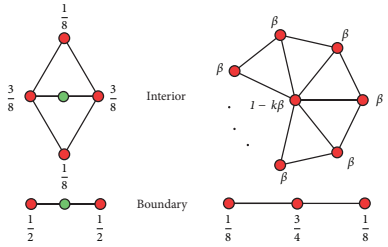

Figure 3: The weights for the new vertex positions in the Loop subdivision scheme. k is the valence (number of incident edges).

The ”old” vertices are calculated by equation (6) in the function VertexRule which is in the LoopSubdivisionMesh.cpp.

2.3 Localize evaluation of the analytical spline

The evaluation of the analytical spline needs only its four neighboring control-points to evaluate the curve. Lots of redundant information can be removed by only looping over the four neighboring control-points instead. This is done by firstly get the newly passed control-point, which is done by flooring t. Now we know the newly passed control-point and can therefore easily fetch the three remaining control-points, the previously passed and the two forthcoming control-points. Now when the four neighboring control-points are known their corresponding basis function needs to be computed at the current position t and then summed together according to equation (7).

$$\boldsymbol{p}(t) = \sum_{i=k-1}^{k+3} \boldsymbol{c}_i b_i(t), k = floor(t) \tag{7}$$

Where $\boldsymbol{c}_i$ is the control-point at i, and $\boldsymbol{b}_i$ is the resulting Bspline value of the cubic basis function at i,see equation (8).

If the one or more of the control-points cannot be fetched, which are the boundary control-points, for example if $k = 0$ then $k - 1 = -1$ which is not acceptable. In these cases the control-points are just ignored.

$$B_{i=0,3}(t) = \\ \frac{1}{6}\begin{cases} [1.2](2+t)^3, & -2 \leq t < -1 \\ -3(1+t)^3 + 3(1+t)^2 + 3(1+t) + 1, & -1 \leq t < 0 \\ -3(1-t)^3 + 3(1-t)^2 + 3(1-t) + 1, & 0 \leq t < 1 \\ (2-t)^3, & 1 \leq t \leq 2 \\ 0, & otherwise \end{cases} \tag{8}$$

Equation (7) was implemented in UniformCubicSpline.cpp in the function GetValue.

2.4 Implement a scheme for adaptive mesh subdivision

The idea is to subdivide the mesh where specifically needed. It could be just random, camera based or in this case mean curvature based subdivision. For more information about how to calculate mean curvature see [2]. With adaptive mesh subdivision there are some rules to follow in order to be able to subdivied a face. Every face has a value true or false which indicates if it should be subdivided or not, and for each face all neighboring faces are checked as well in order to see how to subdivide the face. In figure 4 three different cases are shown where there are at least one false neighboring triangle, and how the triangles that are true should be subdivided in order to prevent holes and subdivision of the false triangles.

As mentioned before the mean curvature was used to adaptively subdivide the mesh more where there curvature is larger than a specific threshold. By doing so the mesh was subdivided more where the surface changes much, for example the nose on the cow, see figure 9.

For each face the mean curvature of the three vertices is calculated by summing the magnitude of the vertices' mean curvature and then divide by three, see equation (9).

$$\boldsymbol{I} = \frac{1}{3}\sum_{i}^{3} abs(\boldsymbol{c}_i) \tag{9}$$

where $\boldsymbol{c}_i$ is the mean curvature of the i:th vertex spanning up the face. Equation (9) was implemented in StrangeSubdivisionMesh.h in the function Subdividable.

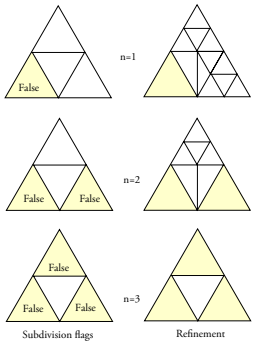

Figure 4: The three possible cases for a triangle that is subdividable (white, true) but has at least one false (yellow) neighbor.

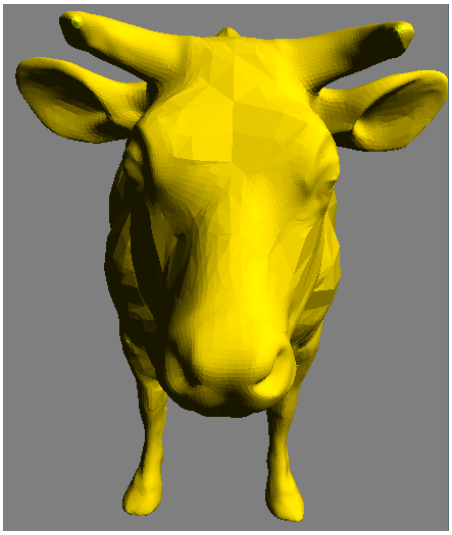

Figure 5: Adaptive subdivision with mean curvature, see especially the nose.

3 Results

3.1 Implement curve subdivision

In figure 6 one can see three steps of curve subdivision. After only three iterations the subdivided curve approximates the analytically curve very well, see figure 6 to the right. This method is faster than the analytically evaluated curve because in the analytically evaluated curve one have to interpolate at each point on the curve the actual value, where in the subdivided curve, it can approximates the curve with less points and therefore be a faster way to represent the curve.

3.2 Implement mesh subdivision

The method for mesh subdivision was done, as mentioned, with the Loop subdivision scheme, see section 2.2 for more details. In figure 7 one can see a before and after subdivision of a cow.

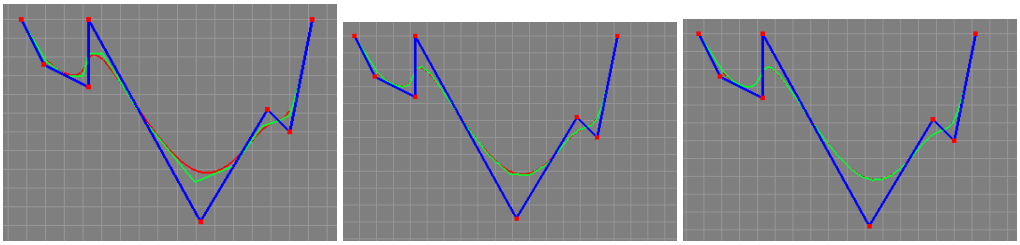

Figure 6: Subdivided curve,green in different stages. From left, subdivision 1, 2, 3. The red curve is the analytically evaluated curve for comparison.

Figure 7: The cow mesh before and after one subdivision iteration.

The cow is now much smoother and looks a lot better and much more like a cow. If one looks close one can still see all the small faces making up the cow. Further subdivision can be made to make these faces smaller and less noticeable to the viewers eyes but is not a recommended method. Because for each triangle four new triangles are created for each old face which will result in a huge amount of triangles after only a few iterations, see equation (10).

$$\boldsymbol{F}_{amount,new} = \boldsymbol{F}_{amount} * 4^i \tag{10}$$

$\boldsymbol{F}_{amount}$ is the amount of faces, and i is the amount of subdivision iterations. Instead of subdivide more steps simple smoothing can be applied instead to get rid of all the small faces. It is a faster and cheaper way to get the desirable result.

3.3 Localize evaluation of the analytical spline

Instead of looping through all the control-points only those control-points which basis function are supported at the given point t is looped though instead, see figure 8. This saves a lot of redundant checking of control-points to see if they give any contribution. By having a localized evaluation of the analytical spline the amount of control-points does not affect the computation time, because it will always only sum no more than the four neighboring control-points at any given point t on the curve. This is a huge improvement if the amount of control-points is large.

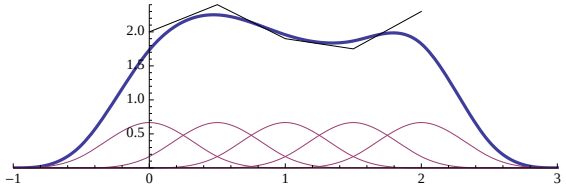

Figure 8: Example of a Bspline and its control-points corresponding basis functions.

3.4 Implement a scheme for adaptive mesh subdivision

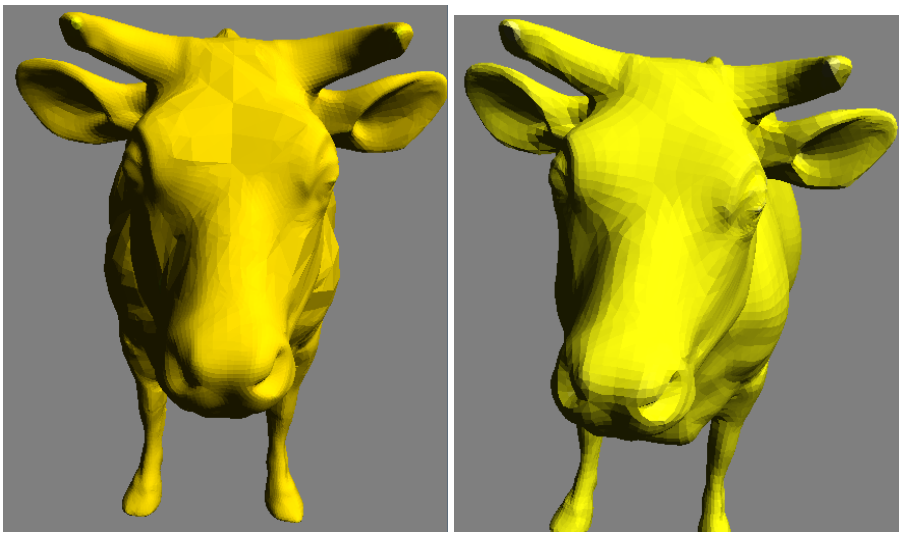

Figure 9: Adaptive subdivision with mean curvature, see especially the nose. Left image is adaptive subdivision and the right image is the Loop subdivision

The adaptive mesh subdivision method chosen for adaptive mesh subdivision was the mean curvature value, where the magnitude of the mean curvature value of each three vertices are added together and then divided by three so a sort of mean value for the face is given. By doing this one can get quite interesting results, see figure 9. Looking at the cows' heads, the flat forehead is not subdivided as much in the left image as the nose is. This is because of the threshold is set so that the faces that have a larger value than the threshold subdivides and the ones with lower value is not subdivided, for instance the flat forehead is not changing as much as the nose and is therefore not subdivided. With this technique we get the subdivision where interesting things happens, where the mesh bends and changes, and no subdivision where subdivision is generally not needed, for example at flat surfaces. This is of course not the only way to do adaptive mesh subdivision but yields quite good results.

4 Conclusion

A good understanding of both spline curves and its different representations such as the analytically evaluation and the iteratively curve subdivision method was gained during this lab. By

reading and implementing subdivision for both curves, as mentioned, and also for meshes gave a very good understanding of how subdivision, the Loop subdivision scheme, works.

Some conclusions can be drawn from these assignments. Here are the conclusions.

Only after a few iterations, three in this case, curve subdivision approximates the analytically spline curve very well, which makes it a very fast and good way of representing a curve.

By localizing the spline curve to only its four neighboring control-points at any given point t the computation of the curve is not dependent on the amount of control-points any more, which makes it faster and redundant computation is removed.

The Loop subdivision scheme works well and yields good result however after each subdivision iteration the amount of faces increases dramatically, see equation (10). Only one or two iterations is needed, in some cases maybe three, and then smoothing can be applied and a very nice and smooth representation of the mesh will be visualized.

There are ways to reduce the amount of faces that are added through subdivision, which is called adaptive subdivision where subdivision is performed where needed. In this lab mean curvature estimate was chosen in order to adaptively subdivide the the mesh where needed, in this case where the mesh has a lot of change in its geometry, for result see figure 9 left.

Lab partner and grade

The lab was carried out by me, Dan Englesson, and my lab partner Emil Brissman. We implemented all mandatory assignments as well as all assignments needed for grade 4 and 5. Therefore me and my lab partner have full-filled the requirements for grade 5.

References

[1] Gunnar Läthén Ola Nilsson Andreas Söderström. Splines and subdivsion. 2011.

[2] Andreas Söderström Gunnar Läthén, Ola Nilsson. Mesh data strucutures. 2011.