LAB REPORT: LAB 3

TNM079, MODELING AND ANIMATION

Viktor Sjögren
viksj950@student.liu.se

Saturday 22nd May, 2021

Abstract

We discuss subdivision methods and spline representations such as Bézier splines and B-splines as ways of elegantly representing smooth curves and surfaces. Uniform cubic splines have been used in an implementation of subdivision of curves. The subdivision segments a curve or line by adding extra control points which is placed and weighted according to the existing points, making each subdivision more and more accurate. A way of smoothing surfaces meshes using Loop's scheme has also been implemented which generates a smoother surface by subdividing every face into smaller faces, with vertices based on the old averages.

1 Background

Subdivision of curves and surfaces are an important part in computer graphics as they allow for fast and efficient computations. The primary method used in the lab was the *uniform cubic B-spline* which stems much from the Bézier curves and general B-splines.

The foundation of subdivision curves also relies on the fact that we parametrize curves. Meaning that each point at a curve will be described as a real number which is linked to a 2D point described by a vector.

The equation for a linear Bezier curve can be seen in equation 1 where two points p_0 and p_1 are the two points which the curve extends between, which is also equal to linear interpolation. The equation sums the coefficients c_0 and c_1 with the basis function $[(1-t), t]$.

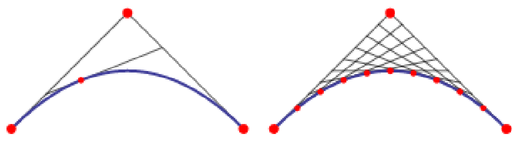


Figure 1: Quadratic Bézier curve, approximating the points on the curve

$$p_1(t) = (1-t)c_0 + tc_1, t \in [0,1] \tag{1}$$

For the Linear case, each point which is traversed by the curve is the same as points as the results of the interpolation. That is not the general case, for example the quadratic Bézier curve which is a combination of two linear curves only goes through the start and end point, approximating the points in between. See Figure 1.

The more relevant curve method in this lab, B-Splines and more notably the most common application of it, the uniform cubic spline is essentially built up by translated copies of a cardinal basis function. This type of spline allows for local control by altering the control points, i.e the coefficients. In a similar manner as the curve points with the quadratic Bézier cure, the control points in a uniform cubic B-spline are instead approximated.

$$B_d(t) = B_{d-1}(s)B_0(t-s)ds \tag{2}$$

The uniform cubic splines are described as seen in equation 2. Note that d is the order of degree which the B-spline is defined for. As the B-splines are constructed by convolutions, obtaining a uniform cubic B-spline means that we apply the convolution step two times.

In the class *UniformCubicSplineSubdivisionCurve.cpp* an implementation of subdivision using uniform cubic spline was made. Given a line which has few sample points the idea was to subdivide each segment into splines. To construct each spline, the coefficients for the control points had to be weighted correctly to ensure that for each subdivision step the subdivision splines more accurately resembles an analytical cubic spline.

The subdivided splines new control points (coefficients), are weighted as in the equations seen in 3

$$\begin{aligned} C'_i &= \frac{1}{8}(c_{i-1} + 6c_i + c_{i+1}) \\ C'_{i+\frac{1}{2}} &= \frac{1}{8}(4c_i + 4c_{i+1}) \end{aligned} \tag{3}$$

With these criterion's in place, it means that the old coefficients will be re-weighted and for each subdivision a new coefficient has to take place between the two old ones. The new coefficient corresponds to $C_{i+\frac{1}{2}}$ in equation 3. It can also be seen that for each re-weighting it is needed information from both sides of the point in interest. This causes the equation mentioned to have boundary problems, meaning it is necessary to have special cases for the points on the boundary. The boundary equation becomes as simple as keeping the old coefficient as they are. See equation 4

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

In the class *LoopSubdvisionMesh.cpp* a subdivision scheme for meshes was implemented. The goal for the implementation was to refine a mesh by adding subdividing each face into smaller faces. This to achieve a smoother surface. The scheme used is the *Loop subdivision scheme*, which does subdivide the faces and computes the new vertex position based on

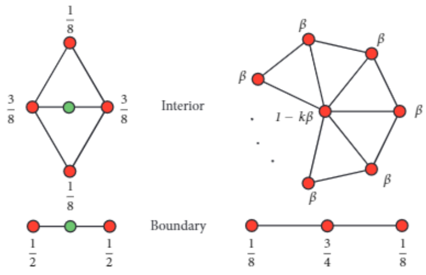


Figure 2: Loop's subdivision scheme, weights for the vertices

their weight. The weight is calculated based on their previous position. In Figure 2, a visual representation of the new weights based on Loop's algorithm can be seen.

The weight value β is obtained through equation 5, where the new position is obtained by summarizing the neighborhood vertices and the original position.

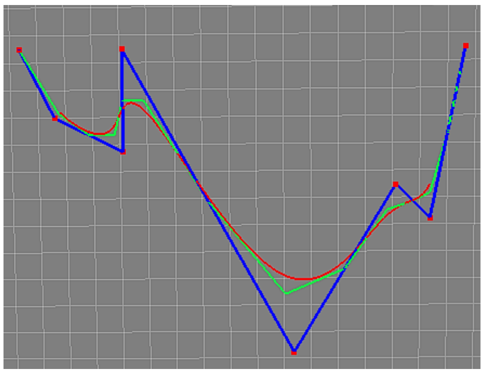
$$\begin{aligned} \beta &= \frac{3}{8k}, k > 3 \\ \beta &= \frac{3}{16}, k = 3 \end{aligned} \tag{5}$$

2 Results

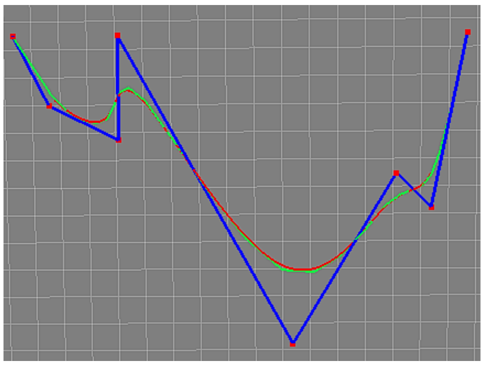
To see the resulting subdivision curve and how effective it resembles an analytical we look at the subdivision curve in three refinement steps and compare the resemblance to the analytical cubic spline.

As can be seen in Figure 3, the analytical cubic spline is visualised in red, while the subdivision curve is marked in green. The blue line segmented by red dots visualises the control points (8 total).

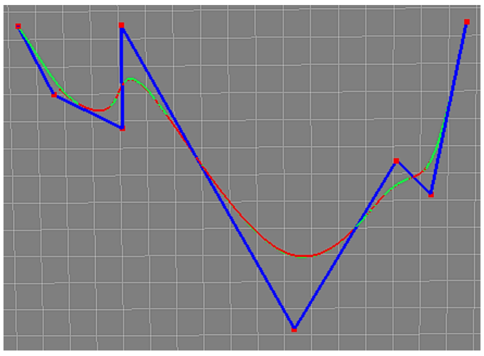
In 3(a) the subdivision curve has been subdivided once, giving the total amount of coefficients that now builds up the splines to be 12. This is correct as we from the beginning start with 8 coefficients, and add one new between each spline segment. In Figure 3(b) another subdivision step has been performed, increasing the amount of coefficients and weighs them so that the splines resemble



(a)



(b)

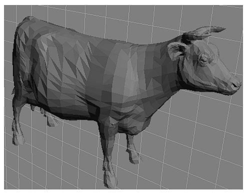


(c)

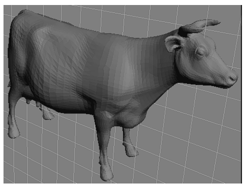
Figure 3: Resulting subdivision curve at different refinement stages. In 3(a) the subdivision curve is only subdivide once, while in 3(c) 3 subdivision has been performed

the analytical one better. After only three subdivision steps, the final subdivision curve is closely similar to the analytical one as seen in Figure 3(c).

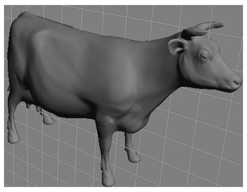
The effects of the Loop's subdivision scheme for meshes can be seen in 4. The resulting mesh becomes vastly more detailed and smooth for each iteration. However also becomes heavier and more computationally complex in return.



(a)



(b)



(c)

Figure 4: Resulting cow mesh, when applying the Loop subdivision scheme onto it. The original cow model (a) contains a total of 5804 faces.

By definition of the scheme, each face is transformed into four faces, meaning we increase the complexity of the model by a rapid pace. After one operation, as seen in Figure 4(b), the model is built up by 23216 faces. For the mesh seen in Figure 4(c), the model contains a face count equivalent to 16 times it's original count, namely 92 864 faces. Because the new faces have coefficients based on their old vertices, the new faces does add in an elegant way a smoother and more realistic look to the meshes. If coefficients were not weighted in any way, or used another simpler scheme, chances are that the mesh subdivision would not be as optimized as with for example Loop's scheme and more faces would be needed to generate the same result in terms of visual fidelity.

3 Lab partner and grade

The lab was made in collaboration with Algot Sandahl. The report scope and content is aimed at grade **3**.

References

[1] M.E. Dieckmann. *Lecture 4,5,6 & 7* , tnm079, 2021.