Splines and Subdivision - Lab 3 TNM079

Jonathan Bosson
jonbo665@student.liu.se

Friday 26th August, 2016

Abstract

The paper discusses how to achieve smooth curves and surfaces on models in computer graphics through B-Splines and subdivision curves. We look at the theory of Bézier curves and basis functions. As well as how to use uniform cubic B-Splines and Loop's Subdivision algorithm to achieve smooth curves.

1 Introduction

Subdivision curves and surfaces have been known for a long time and are used in applied computer graphics in numerous ways. Interaction with the splines is easy since all operations only impact a local area. At the same time the computation cost is low since the order of the splines can be kept relatively low. The numerical scheme is stable because all computations are on stable bases. This paper only focuses on the assumption of uniform basis functions, which allows for a simpler analysis and construction however it limits some of the geometric properties.

2 Background

Bézier curves were developed in the late 50s as a stable scheme to approximate parametric curves by Paul de Casteljau. Later in the 60s an engineer named Pierre Bézier designed car bodies using the algorithms from where it bears its name. Bézier curves can be seen as a sum of different coefficients (c_0, c_1, etc) using the basis functions $\{(1-t), t\}$.

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

$$\begin{aligned} p_2(t) &= (1-t)p_{0,1}(t) + tp_{1,1}(t) \ , t \in [0,1] \\ &= (1-t)^2 c_0 + 2t(1-t)c_1 + t^2 c_2 \ , t \in [0,1] \end{aligned} \tag{2}$$

The simplest version, a linear Bézier curve with points p_0 and p_1 is just a straight line (equivalent to linear interpolation), equation given by 1. Quadratic Bézier curves however takes the form of the path traced by c_0, c_1 and c_2 and is a combination of two linear Bézier curves as seen in equation 2.

Looking at the basis functions, each coefficient only supports the curve locally meaning that only a few of the functions are non-zero. Given a function with a set of points $b_i(t)$ can the function value be generated by calculating the sum.

$$p(t) = \sum_i c_i b_i(t) \tag{3}$$

The most commonly used B-Spline is the Uniform Cubic B-Spline which is a cubic polynomial meaning it will have three coefficients (also called control points) that supports each point. Each control point is approximated but can be manipulated to achieve the desired curve.

$$B_d(t) = \int B_{d-1}(s) B_0(t-s)\, ds \tag{4}$$

B-Splines can be defined as the convolution of the lower order B-Spline with the lowest coefficient,

like equation 4. The order of the function is given by the length of the intervals where both functions are non-zero. As the current B-Spline is calculated using the previous order is a recursive computation required. Defining the B-Splines as convolutions allows for a refinement equation, 5, which in turns connects it to subdivision methods.

$$B_d(t) = \frac{1}{2^d} \sum_{i=0}^{d+1} \binom{d+1}{i} B_d(2t - i) \tag{5}$$

where the binomial is

$$\binom{k}{m} = \frac{k!}{m!(k-m)!} \tag{6}$$

The refinement equation states that a B-Spline of degree d is equal to a sum of translated (i) and dilated ($2t$) copies of itself. The resulting refined cubic B-Spline of order 3 will then be

$$\begin{aligned} B_3(t) = \frac{1}{8}(\quad & 1B_3(2t) \\ + \quad & 4B_3(2t-1) \\ + \quad & 6B_3(2t-2) \\ + \quad & 4B_3(2t-3) \\ + \quad & 1B_3(2t-4) \\ ) & \end{aligned} \tag{7}$$

This is used in subdivision as we can express the curve in B-Splines whose support is twice as dense but half as wide as in equation 8.

$$p(t) = B(t)C = B(2t)SC \tag{8}$$

The coefficients C_i are thus redefined as $C_{i+1} = SC_i$. S is defined from equation 5 or directly as

$$s_{2i+k,i} = s_k = \frac{1}{2^d} \binom{d+1}{k} \tag{9}$$

where i is the row, k is the column, and d is the degree. The subdivision will not yet work on the boundary as it is a weighted average of coefficients on both sides of current point. To solve this specific rules to subdivision on the boundary are provided. The normal solution to this is to define different spline basis functions at the boundary such that the sum still becomes 1. This is especially convenient in situations where efficiency is important as it can rescale the splines at the boundary with scalar values. The rules used in the lab was done in *Lane and Riesenfeld* [2] ang gives following boundary values

$$\begin{aligned} & s_{0,0} = 1 \\ & s_{1,0} = .5 \qquad s_{1,1} = .5 \\ & s_{n-1,m-1} = .5 \quad s_{n-1,m} = .5 \\ & s_{n,m} = 1 \end{aligned} \tag{10}$$

As well as approximating a curve by successive applications of subdivision of the uniform cubic splines can the same be done for meshes. One method of this is through Loop's subdivision scheme [3]. It uses a straightforward approach and is easy to implement with access to face and vertex neighborhood. Each original triangle is divided into four new triangles with the new vertex positions as weighted averages of the original surrounding ones in its 1-ring with β as weight, k is the valence of current vertex point.

$$\beta = \begin{cases} \dfrac{3}{8k}, & k > 3 \\ \dfrac{3}{16}, & k = 3 \end{cases} \tag{11}$$

3 Results

Although this lab went through a lot of theory the implementations were fairly straightforward and easy to compute. Coding went through two seperate files, `UniformCubicSplineSubdivisionCurve` and `LoopSubdivisionMesh`.

3.1 UniformCubicSplineSubdivisionCurve

In `UniformCubicSplineSubdivisionCurve` the task was to find the new coefficients C after a subdivision had been made as well as make sure the subdivision doesn't break any of the boundary rules. The first new coefficient is an average of the two original control points. The last control point is simply defined as the last control point in the initial list of coefficients. All other coeffiecients were defined weighting the three nearby control points and adding a

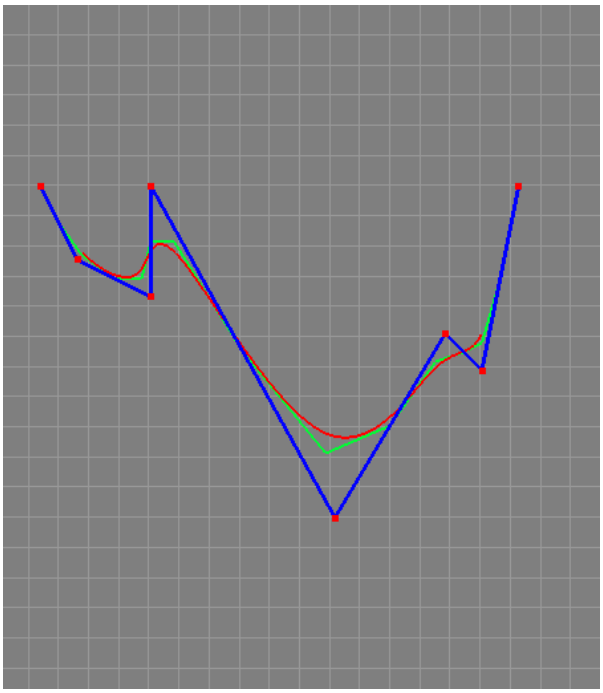

Figure 1: Green Uniform Cubic B-Splines subdivided curve converges towards the analytical red

new control point weighted by current and the next control point. Or in more simpler terms

$$\begin{aligned} C'_0 &= \tfrac{1}{2}(C_0 + C_1) \\ C'_j &= \tfrac{1}{8}(C_{i-1} + 6C_i + C_{i+1}) \\ C'_{j+1} &= \tfrac{1}{2}(C_i + C_{i+1}) \\ C'_{n-1} &= C_{n-1} \end{aligned} \tag{12}$$

where C' is twice as large as C and n is the size of the coefficient list.

Figure 1 shows how the green curve created through uniform cubic splines converges to the analytical red curve as the subdivision scheme is run more and more.

3.2 LoopSubdivisionMesh

The task in `LoopSubdivisionMesh` was to implement the weighting of the vertex rules of the new vertex points. For each vertex, the new position v was defined as the a sum of all neighbouring vertex positions weighted by the β from 11 as well as its original position weighted by $(1 - k\beta)$, where k is the valence.

$$\vec{v}' = (1 - k\beta)\vec{v} + \sum_i \vec{v}_i \beta \tag{13}$$

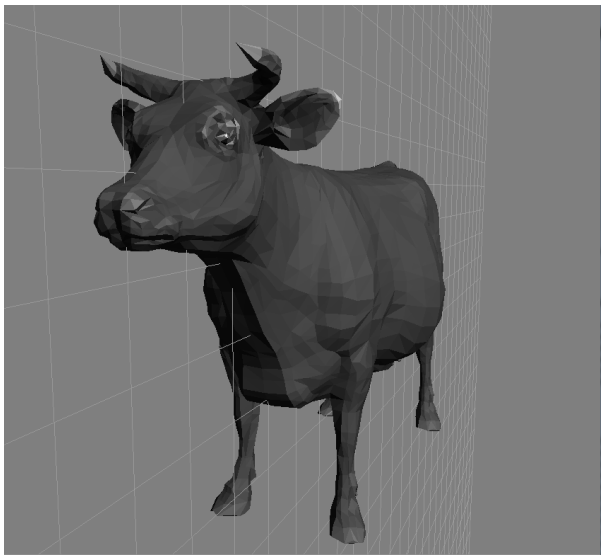

Figure 2: A cow model in it's original resolution

The result can be seen in figure 2, 3 and 4 where the detail of the cow model increases vastly. With each subdivision is the resolution of the mesh quadrupled.

References

[1] M. E. Dieckmann, *Lecture Slides for TNM079*, **Lecture 4**, 2016.

[2] J.M. Lane and R.F. *Riesenfeld. A theoretical development for the computer display and generation of piecewise polynomial surfaces.* In IEEE Transactions PAMI2, 1980.

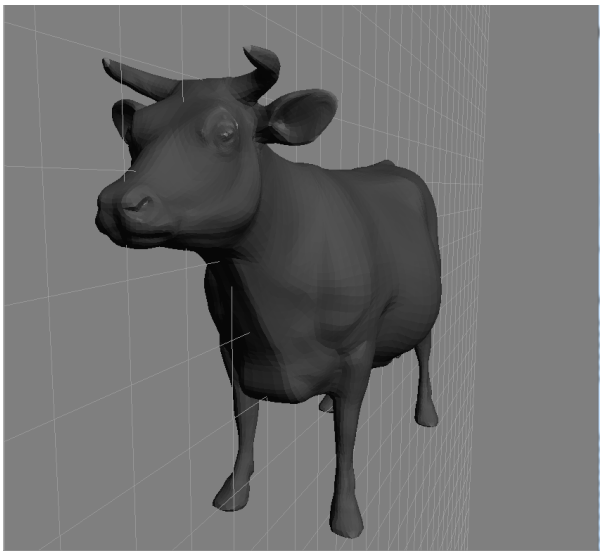

Figure 3: Model subdivided once with Loop's algorithm
b

[3] C. Loop. *Smooth subdivision surfaces based on triangles. Masters thesis, Department of Mathematics*, University of Utah, August 1987

This report aims for grade 3.

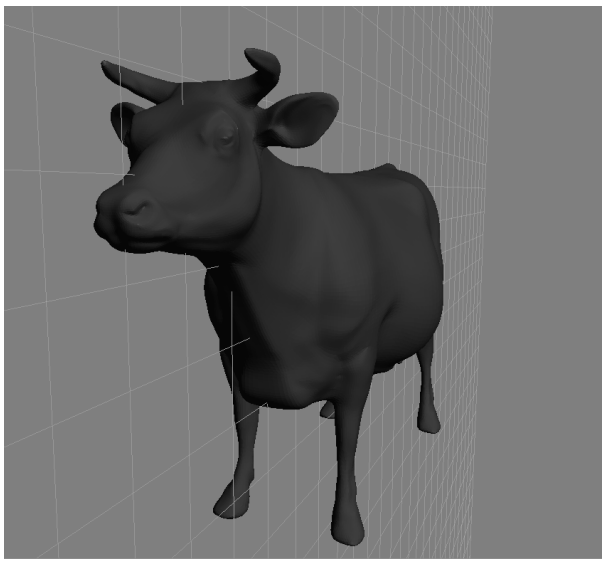

Figure 4: Model subdivided twice with Loop's algorithm