LAB REPORT: LAB 3

TNM079, MODELING AND ANIMATION

Lucas Hägg
lucha567@student.liu.se

Monday 10th June, 2024

Abstract

In this lab, an implementation of mesh data structures and curve subdivision methods were done. The main goal was to understand and improve how to use methods for creating smoother curves and meshes in graphics. Techniques used were Loop subdivision for mesh and uniform cubic spline for curves. These methods help to make smoother transitions and more details in both curves and 3D mesh models.

1 Introduction

The specific purpose of this lab is to explore and implement loop subdivision for meshes and the uniform cubic spline for curves. In computer graphics, the ability to refine and smooth out models is crucial for the realism and quality of visual representations. By understanding and applying these subdivision techniques, one can achieve more precise and visually appealing results.

2 Background

The main focus was on the practical application of two specific subdivision techniques: The uniform cubic spline subdivision for curves and Loop subdivision for meshes.

2.1 Uniform Cubic Spline Subdivision for Curves

This process involves breaking down a given curve into smaller segments and applying a mathematical formula to smooth out the curve. Every existing control point C_i is replaced with two new points:

$$C_{\text{new}} = \frac{1}{8}(c_{i-1} + 6c_i + c_{i+1}) \tag{1}$$

$$C_{\text{new next}} = \frac{1}{8}(4c_i + 4c_{i+1}) \tag{2}$$

2.2 Loop Subdivision for Meshes

The first task involved implementing the Loop subdivision technique to refine mesh structures. This method is highly regarded for its ability to generate smoother surfaces by subdividing the input mesh into smaller triangles. Each vertex and edge within the mesh was recalculated to produce a finer mesh with improved smoothness and detail. The process was done by first identifying each vertex in the mesh, then recalculating its position based on neighboring vertices. This position was determined by two rules, Vertex Rule and Edge Rule.

Vertex Rule:
The vertex rule recalculates the position of an existing vertex based on its neighboring vertices.

1. Get all neighbor vertices and store the amount in a valence variable.

2. Calculating the Beta Coefficient

   Beta(valence) determines the smoothing weight based on the valence of the vertex. This weight adjusts how much influence the neighboring vertices have on the new position of the current vertex. The weight is lower for vertices with more neighbors, reducing the influence any single neighboring vertex has.

$$\beta(\text{valence}) = \begin{cases} \frac{1}{16}, & \text{if valence} = 3 \\ \frac{3}{16}, & \text{if valence} = 3 \\ \frac{3}{8 \times \text{valence}}, & \text{otherwise} \end{cases} \tag{3}$$

3. Calculate the new position

   The position of the vertex is calculated as a weighted average of its original position and the positions of its neighbors.

$$V_{\text{new}} = V_{\text{old}} \times (1 - n \times \beta) + \sum_{i=1}^{n} (\text{pos}_i \times \beta) \tag{4}$$

   where n is the valence, β is the weighting factor and pos_i are the positions of the neighboring vertices.

Edge Rule:
The edge rule calculates the position of a new vertex along an edge in the mesh. The function considers the vertices v0 and v1 on both sides of the edge, as well as the vertices connected to v0 and v1 but not lying on the primary edge. These are the vertices connected to v0 and v1 via adjacent edges. The new vertex is placed along the edge using a specific weighted formula that involves all four vertices:

$$V_{\text{new}} = \left(\frac{3}{8} \times (v0 + v1)\right) + \left(\frac{1}{8} \times (v2 + v3)\right) \tag{5}$$

3 Tasks

This chapter goes through how the tasks were implemented during the lab.

3.1 Subdivide

The **Subdivide()** function subdivides a curve. The data structure has a vector which stores the current control points V_{curr}. The function was implemented as following:

1. Create an empty vector V_{new} to store the new control points.
2. Add the original first control point to V_{new}, keeping the same start point.
3. Add the first new subdivided point from equation 2.
4. Loop Over the remaining control points and add two new control points from equation 1 and 2 to V_{new} until the very last control point in V_{curr}.
5. Add the last control point in V_{curr} to V_{new}.

3.2 Vertex rule

The **VertexRule()** function started by accessing the vertex position from the input index, using a function to retrieve the vertex based on index. Then, calling an existing function to get the neighboring vertices. The Valence was determined from the length of the neighbor list. The β value was calculated from function 3. The new vertex position was then calculated according to equation 4

3.3 Edge rule

The **EdgeRule()** function was implemented by first accessing the needed vertices. Using the half-edge data structure, the positions of the two vertices at the ends of the edge were stored as v_0 and v_1. Then finding v_2 and v_3 by accessing the half-edges previous edges, again using the half-edge data structure. The new vertex position was then calculated using equation 5.

4 Results

First are the results for curve subdivision:

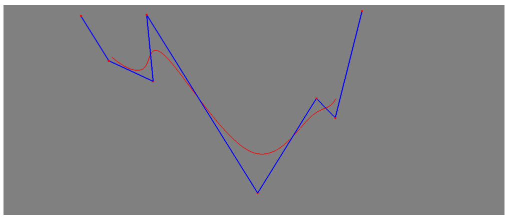

Figure 1: Curve without any subdivision. Blue line is starting point and red line is end goal.

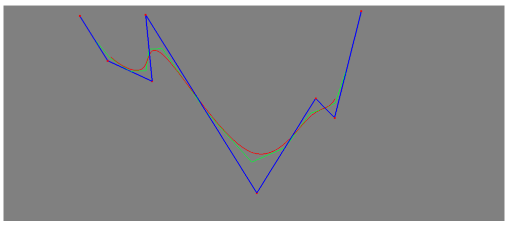

Figure 2: Curve subdivided once. The curve is seen in light green.

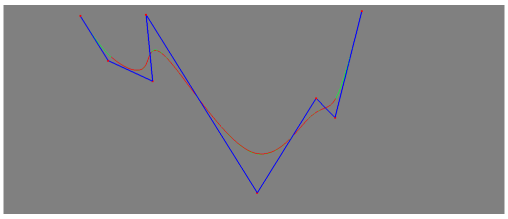

Figure 3: Curve subdivided three times, almost equal to the red line.

Next are the results of the loop subdivision for a mesh cube:

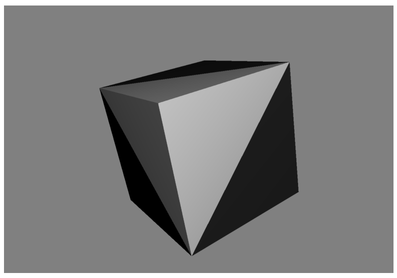

Figure 4: Cube starting point.

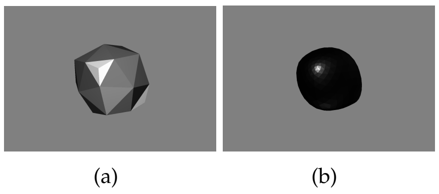

Figure 5: (a) Sphere eroded with 30 iterations, (b) Sphere eroded with 30 iterations, then 30 iterations of dilation.

5 Conclusion

This lab demonstrated the effectiveness of Loop subdivision for meshes and uniform cubic spline for curves in enhancing graphical smoothness and detail.

6 Lab partner and grade

Lab partner was Kenton Larsson and the grade I am for is 3.

References

[1] Mark Eric Dieckmann. *MESH DATA STRUCTURES*. Linkoping university, 2023.