CSC4140 Assignment V

Computer Graphics

April 5, 2022

Geometry

This assignment is 9% of the total mark.

Strict Due Date: 11:59PM, Apr 5^{th}, 2022

Student ID: 118010335

Student Name: Wei WU

This assignment represents my own work in accordance with University regulations.

Signature:

1 Overview

This project has two parts. The first part is about evaluation of Bezier curves and surfaces using de Casteljau's algorithm. The second part is about triangle meshes. To be specific, the computation of area-weighted vertex normal, the edge flip and edge split operation, and the loop subdivision.

2 Part I Bezier Curves and Surfaces

2.1 Bezier curves with 1D de Casteljau subdivision

De Casteljau's algorithm calculates the weighted average of two control points level by level. In each iteration, the number of control points is reduced by 1. In the last level, there is only one control point left, which is the final point on the Bezier curve corresponding to parameter t.

Bernstein form of a Bézier curve of order n:

$$\mathbf{b}^n(t) = \mathbf{b}_0^n(t) = \sum_{j=0}^{n} \mathbf{b}_j B_j^n(t)$$

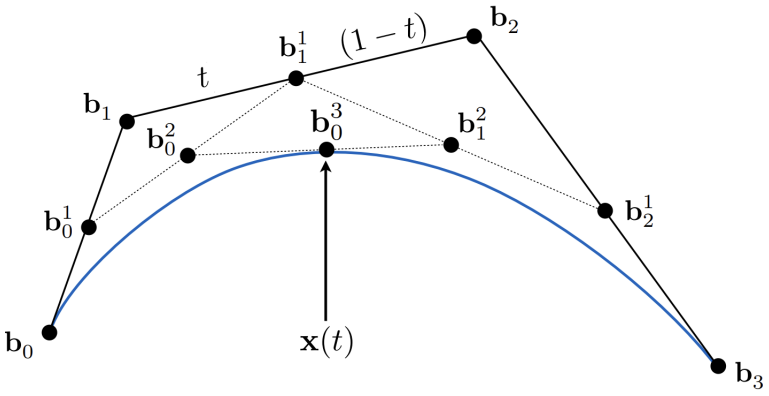
Bézier curve order n
(vector polynomial of degree n)

Bernstein polynomial
(scalar polynomial of degree n)

Bézier control points
(vector in R^N)

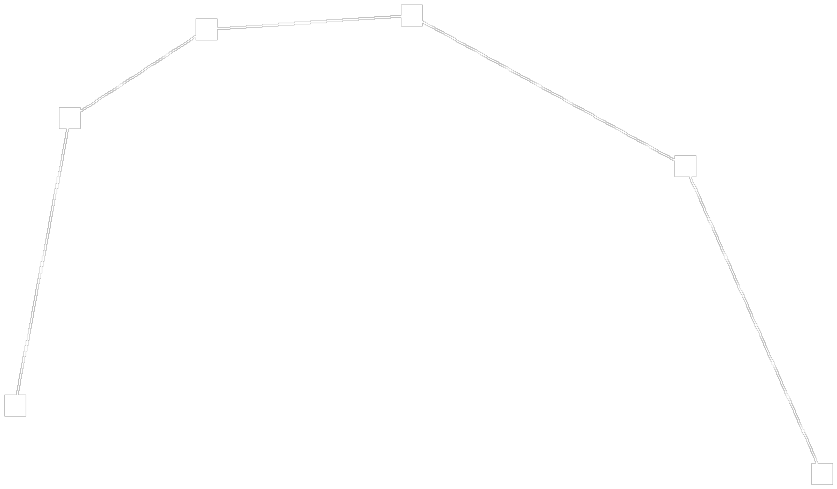
Bernstein polynomials:

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

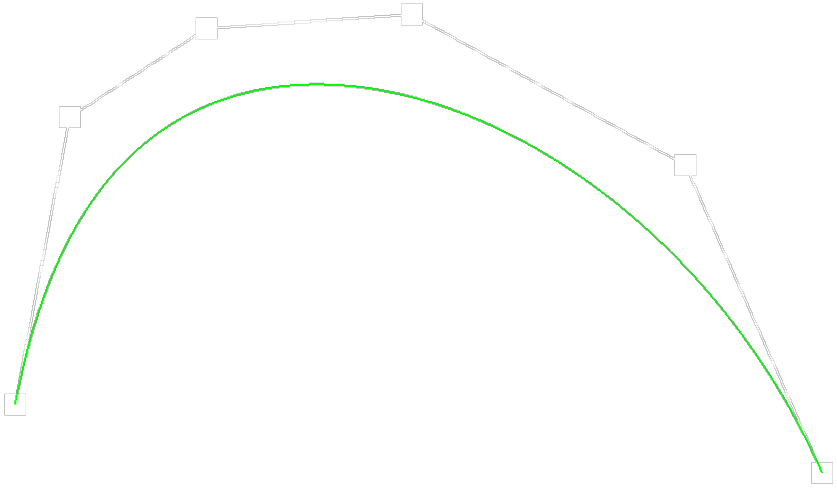


My own Bezier curve with 6 control points is bzc/my_curve.bzc.

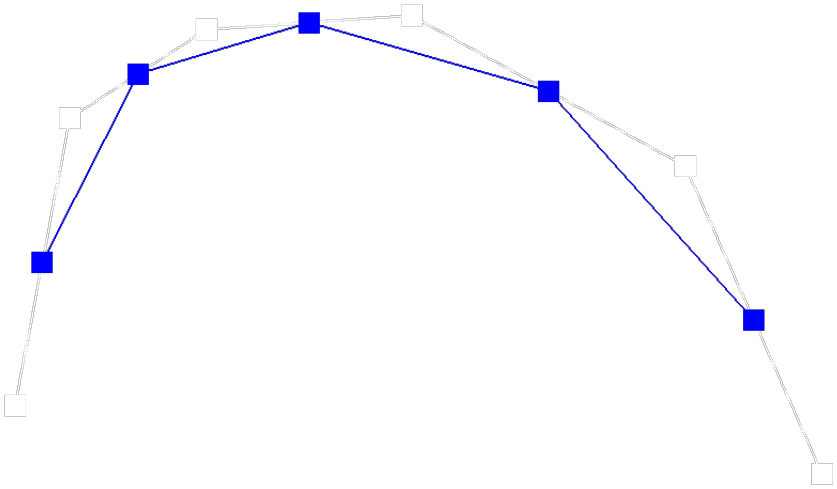
Screenshots of each step / level of the evaluation from the original control points down to the final evaluated point:



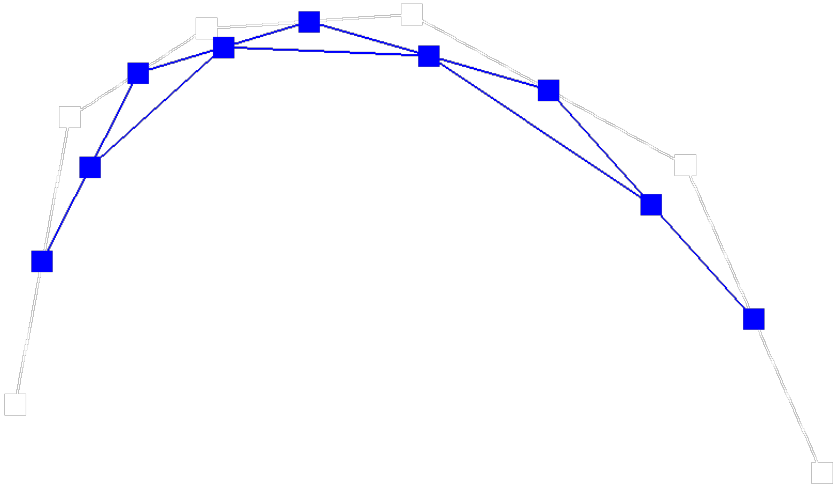
Framerate: 462 fps



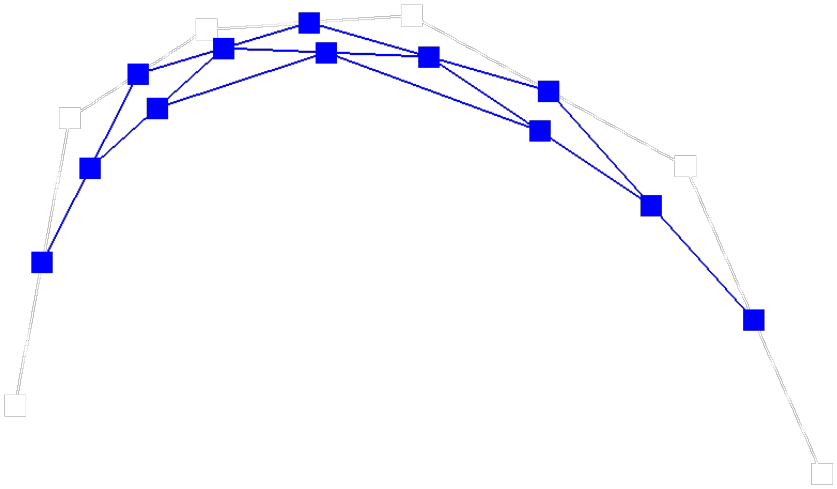
Framerate: 441 fps



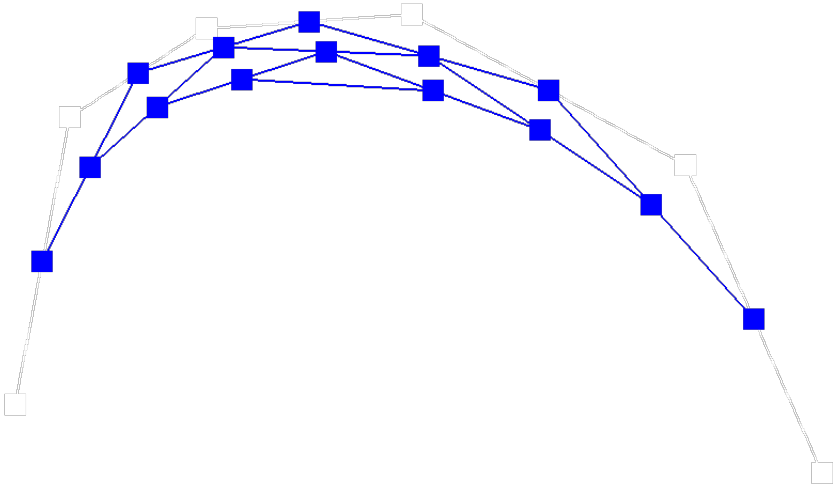
Framerate: 461 fps



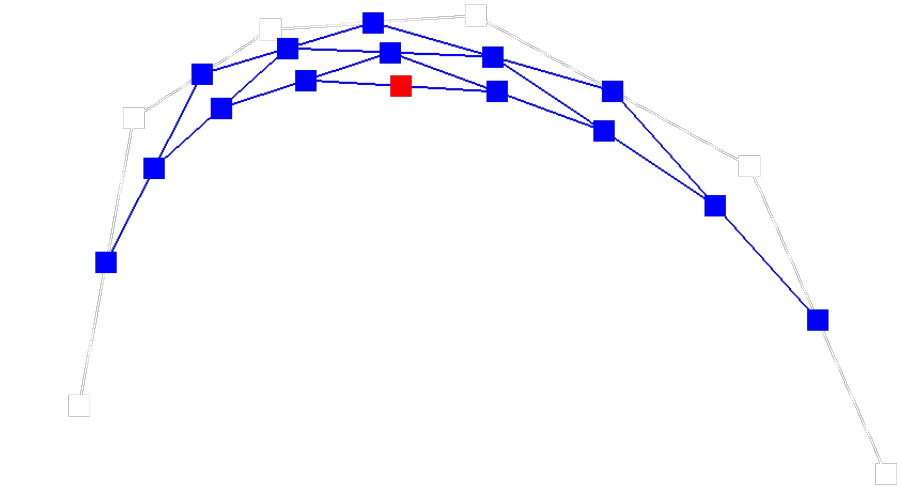
Framerate: 451 fps



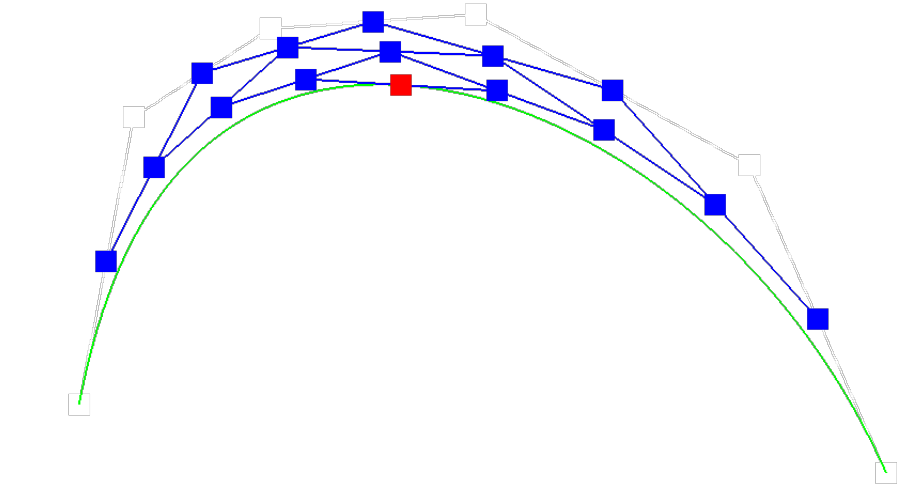
Framerate: 428 fps



Framerate: 431 fps

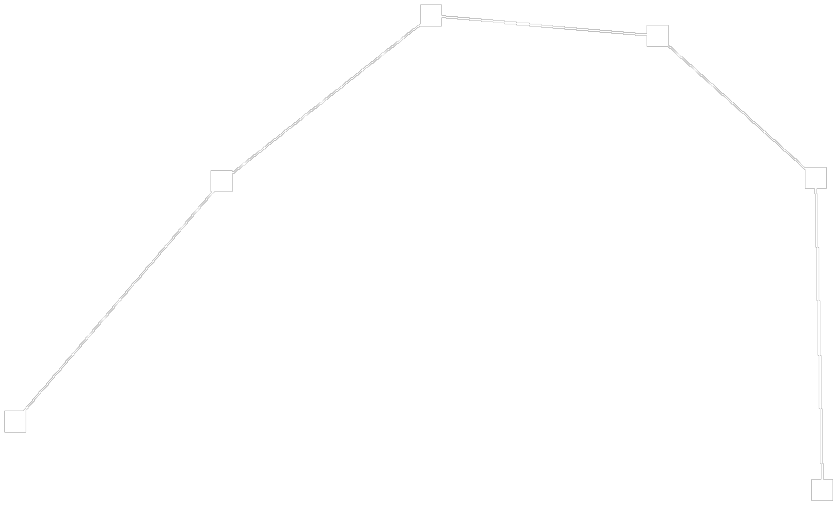


Framerate: 422 fps

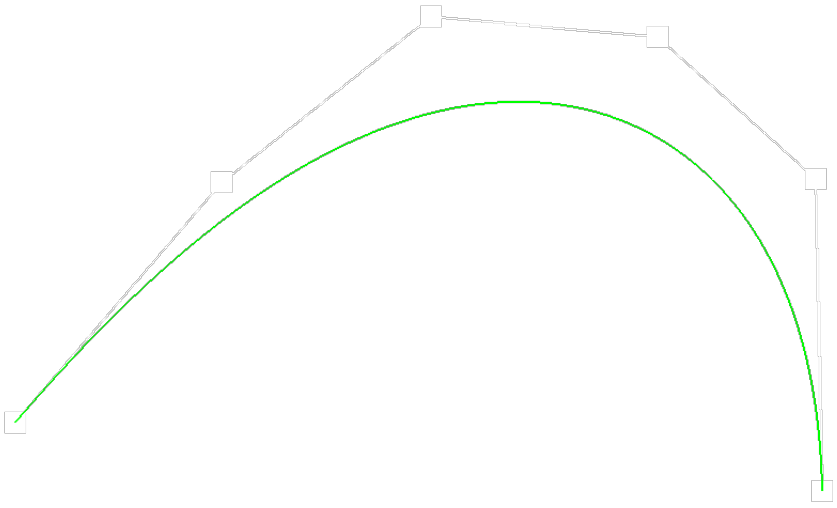


Framerate: 414 fps

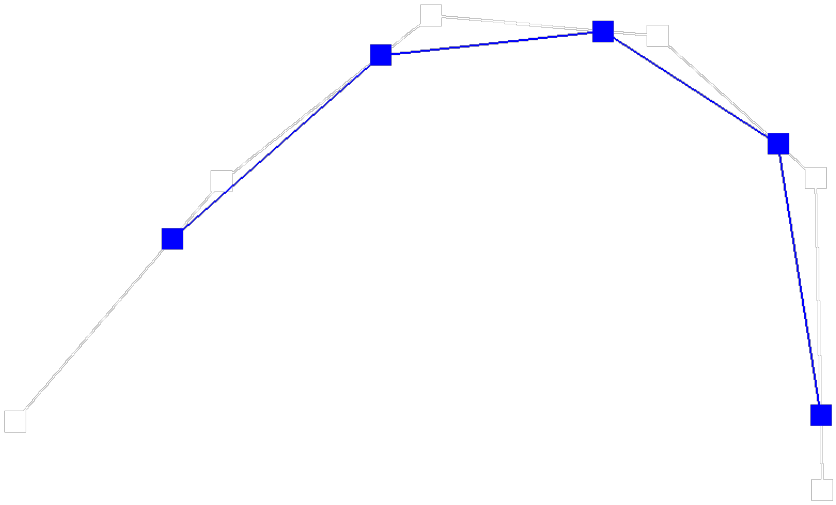
Screenshots of a slightly different Bezier curve by moving the original control points around and modifying the parameter t via mouse scrolling



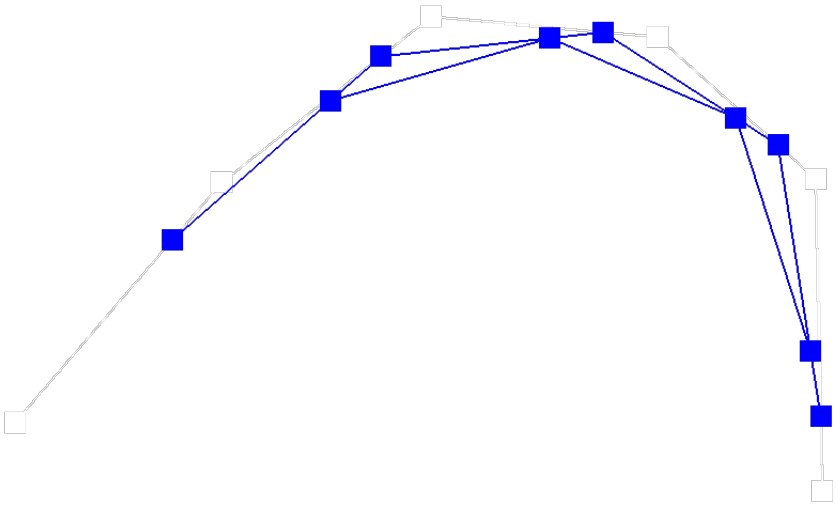
Framerate: 424 fps



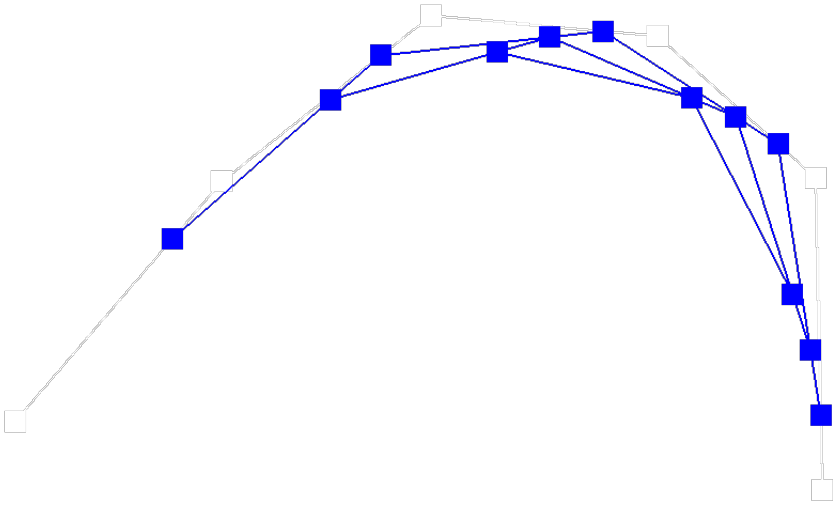
Framerate: 428 fps



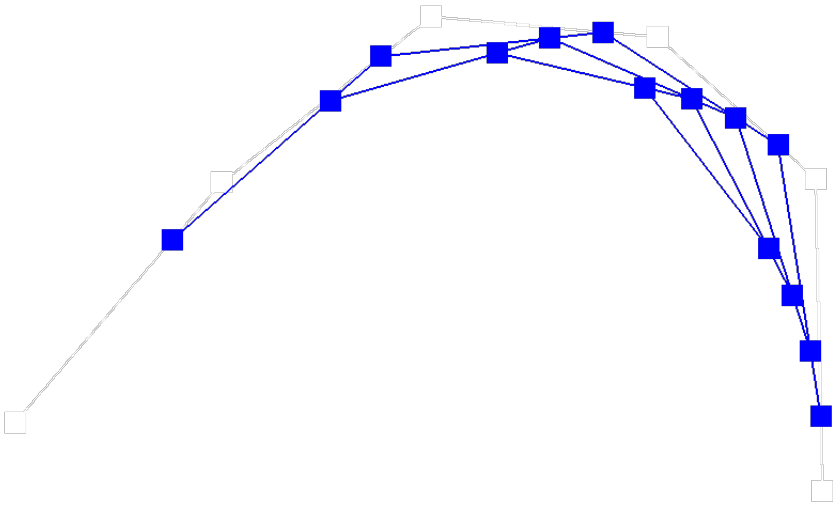
Framerate: 453 fps



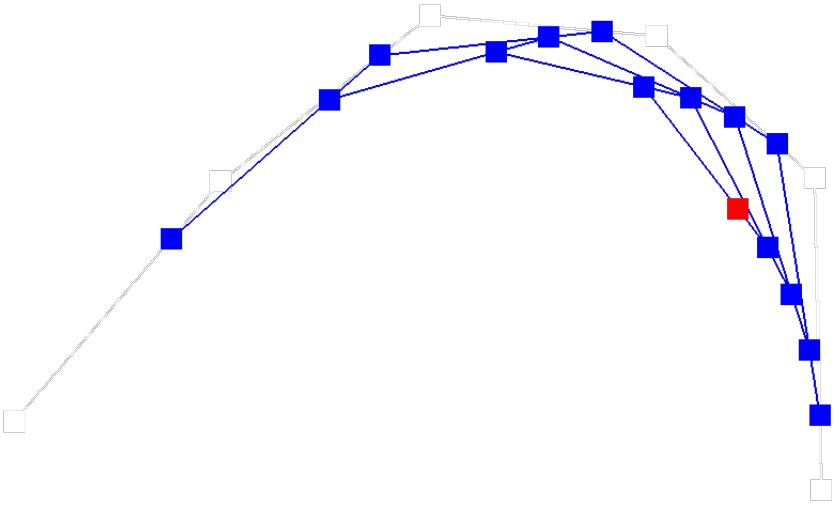
Framerate: 446 fps



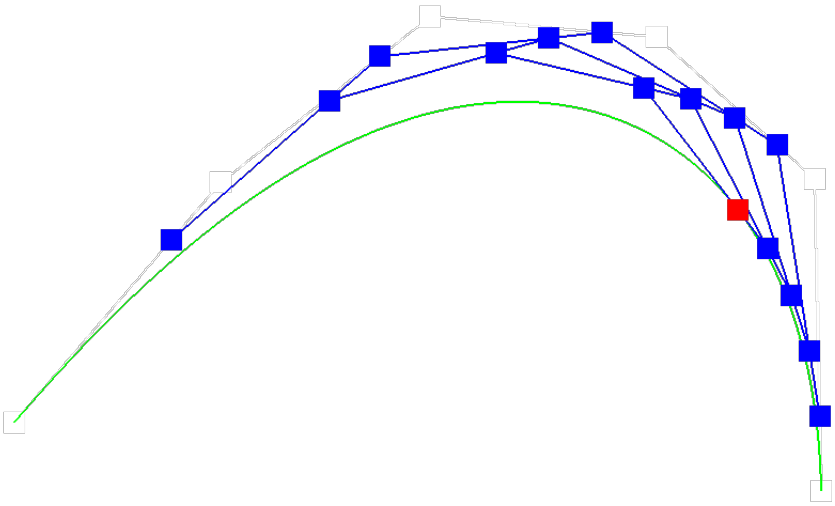
Framerate: 450 fps



Framerate: 431 fps



Framerate: 437 fps

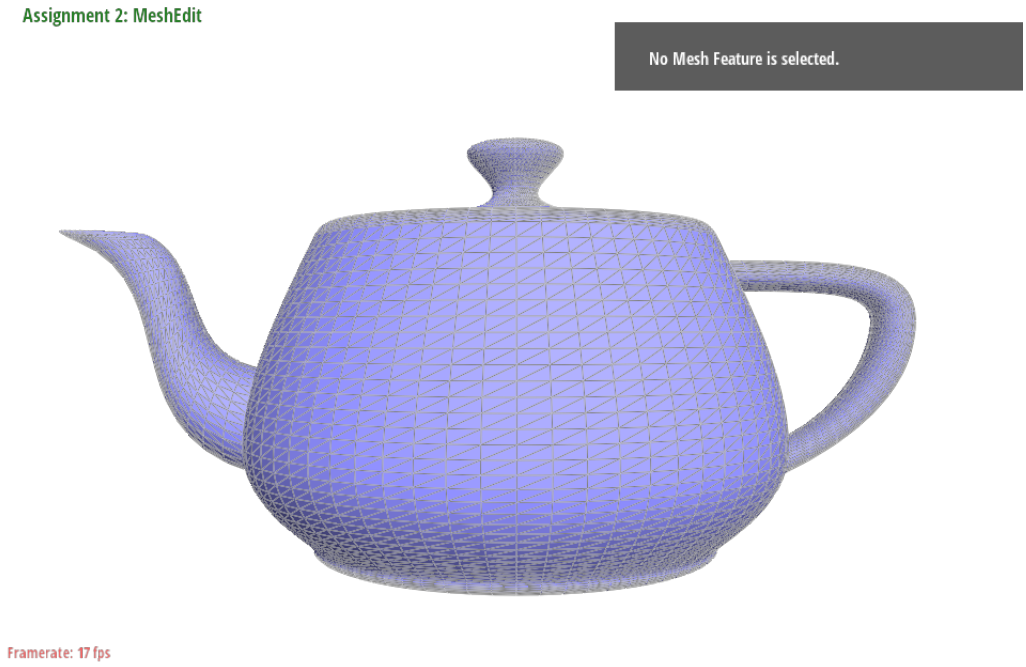


Framerate: 409 fps

2.2 Bezier surfaces with separable 1D de Casteljau

For Bezeir surfaces, we run de Casteljau's algorithm for two rounds. In the first round, we run de Casteljau's algorithm row by row, with respect to parameter u, getting a column of control points. In the second round, we run de Casteljau's algorithm for the column, with respect to parameter v. The final point is a point on the Bezier surface, corresponding to parameter (u, v).

A screenshot of bez/teapot.bez (not .dae) evaluated by your implementation:



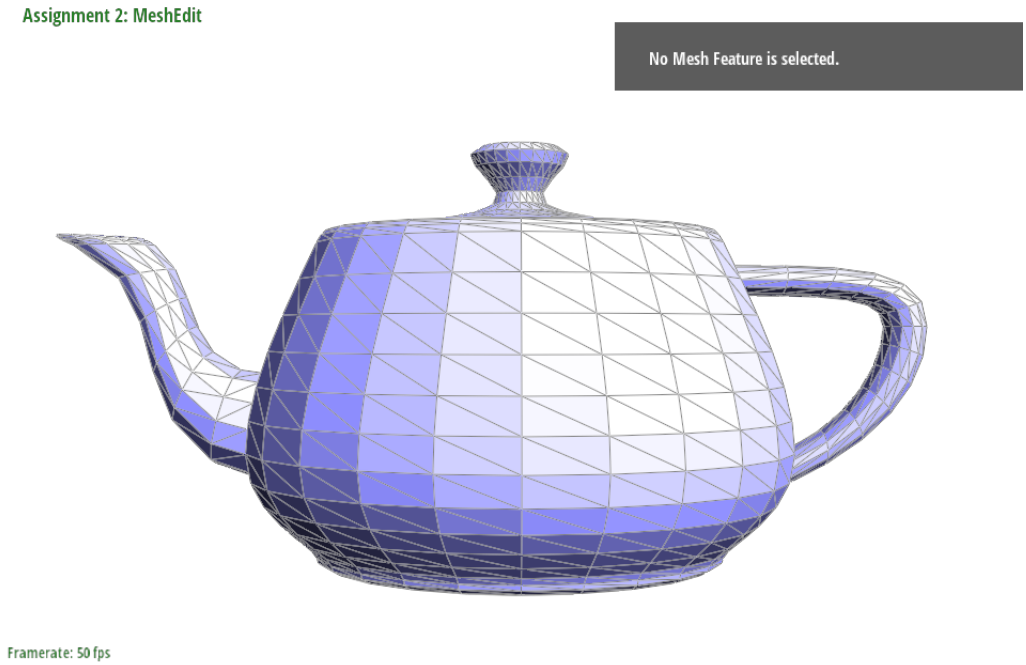
3 Part II Bezier Curves and Surfaces

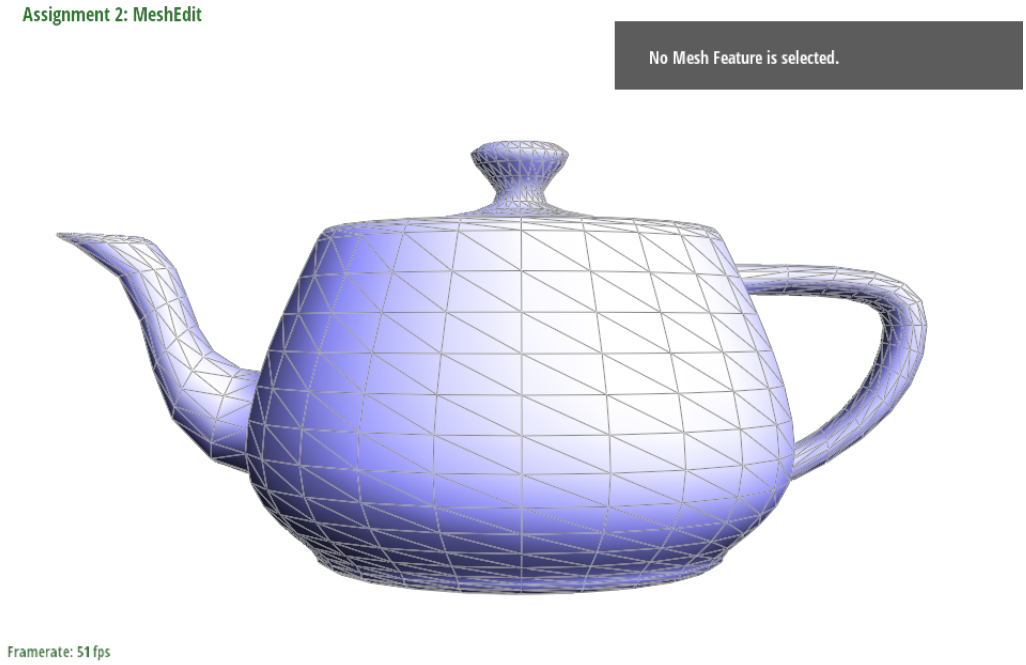
Triangle meshes are explicit representation. At rasterization, the triangles are firstly projected onto the render buffer. Then, it is easy to check whether a pixel is inside a triangle. While Bezier surfaces are implicit representation. At rendering time, it is difficult to check whether a pixel is on the surface. If so, it is also complicated to calculate the uv coordinates. Therefore, Bezier surfaces are much more difficult to render directly.

3.1 Area-weighted vertex normals

For each face containing the vertex, we calculate the product of its area and normal vector, which is equal to the cross product of two of its halfedges. Then, we sum all the cross products up. Finally, we normalize the sum to get the vertex normal.

Screenshots of dae/teapot.dae (not .bez) comparing teapot shading with and without vertex normals:

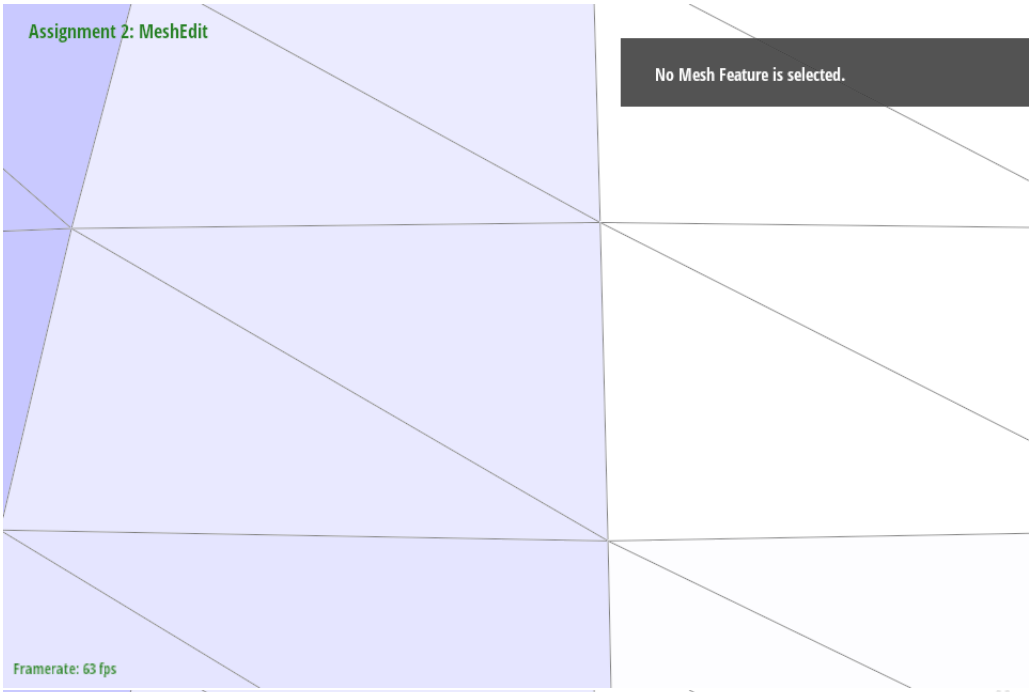


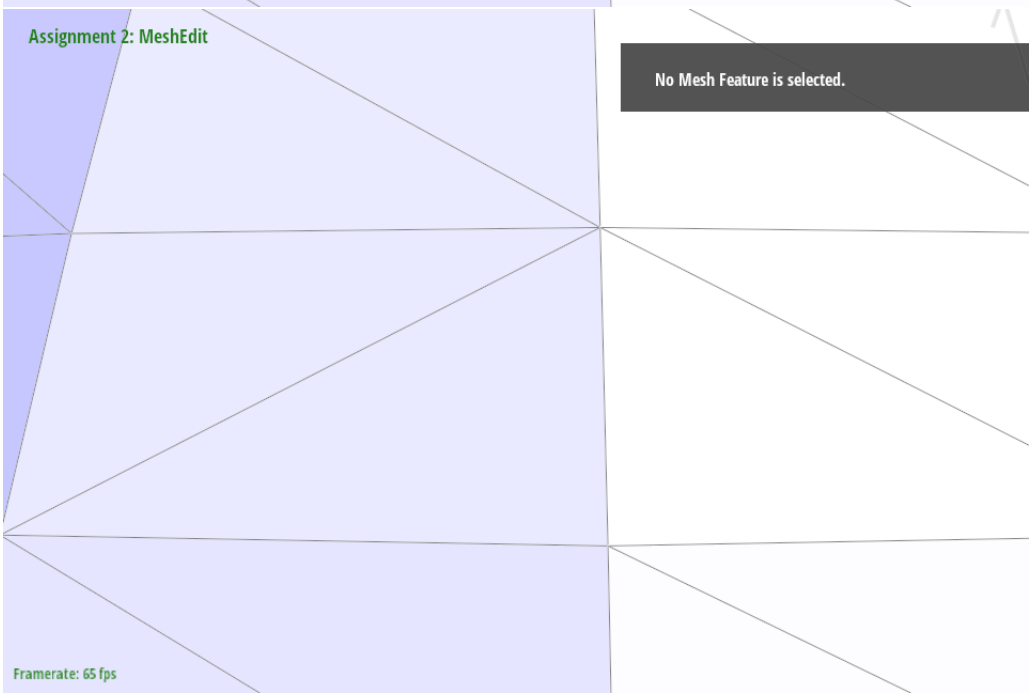


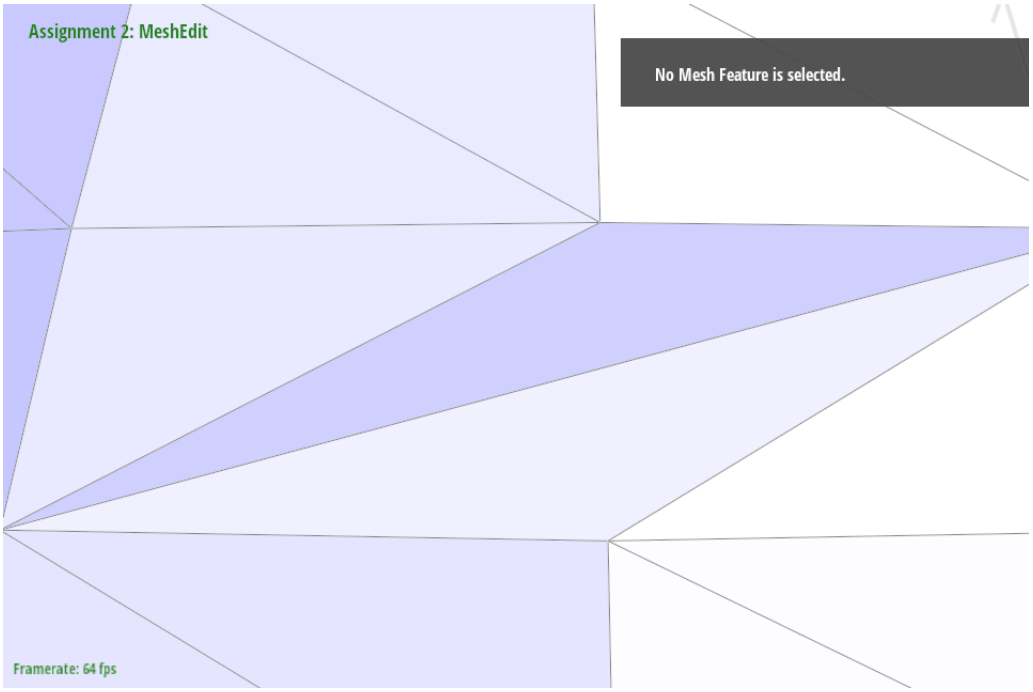
3.2 Edge flip

In edge flip operation, the number of vertices, halfedges, edges, and faces do not change. Hence, we firstly record all the related elements, then modify each element's pointers to other elements.

Screenshots of the teapot before and after some edge flips:


Assignment 2: MeshEdit
No Mesh Feature is selected.
Framerate: 63 fps


Assignment 2: MeshEdit
No Mesh Feature is selected.
Framerate: 65 fps



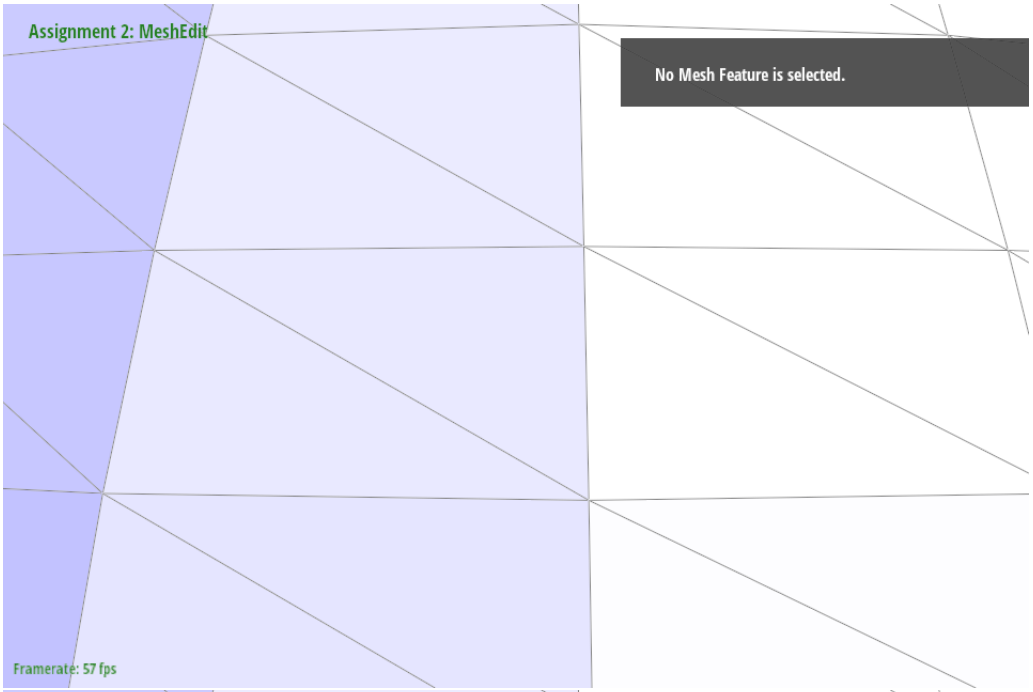
3.3 Edge split

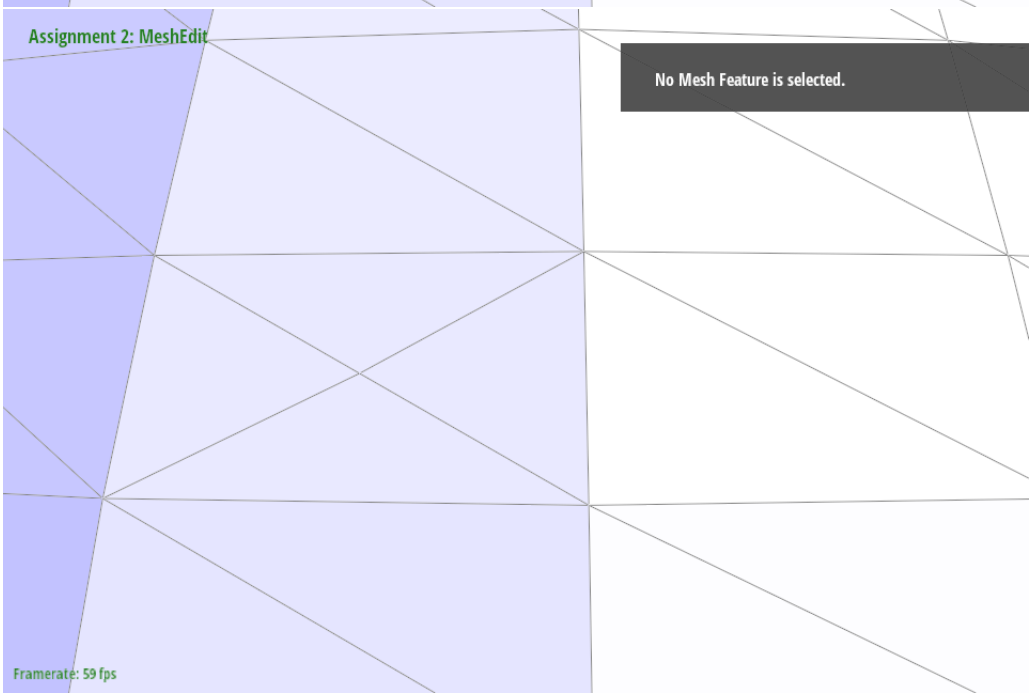
If we flip an edge on the boundary, we get an edge connecting a vertex of the physical mesh and a vertex of the virtual boundary face. First, the virtual boundary face may not be a triangle, which means we do not know which one of its vertices to connect. Even if we know which vertex to connect, each of the resulting two faces is composed of both physical mesh and virtual boundary, which makes no sense.

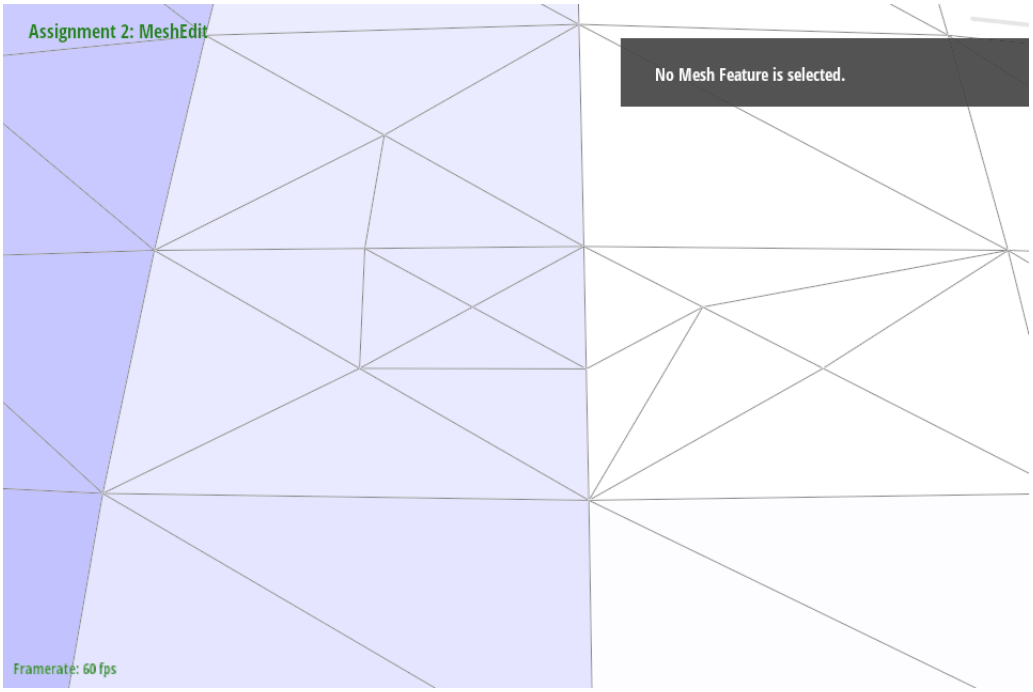
On the other hand, if we split an edge on the boundary, we add a new vertex as well as a new edge connecting it to a vertex of the physical mesh. The resulting two faces are totally physical. Hence, splitting a boundary edge does make sense.

In edge split operation, the number of vertices, halfedges, edges, and faces increase. Therefore, we first create new elements. Then, similar to the edge flip operation, we modify each element's pointers to other elements.

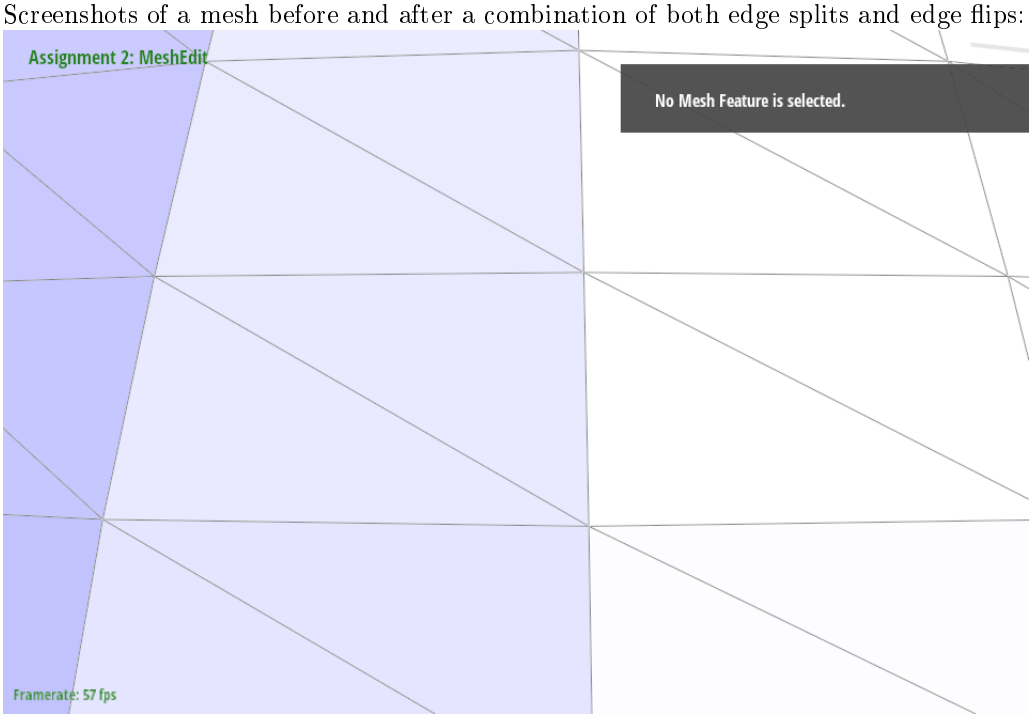
Screenshots of a mesh before and after some edge splits:

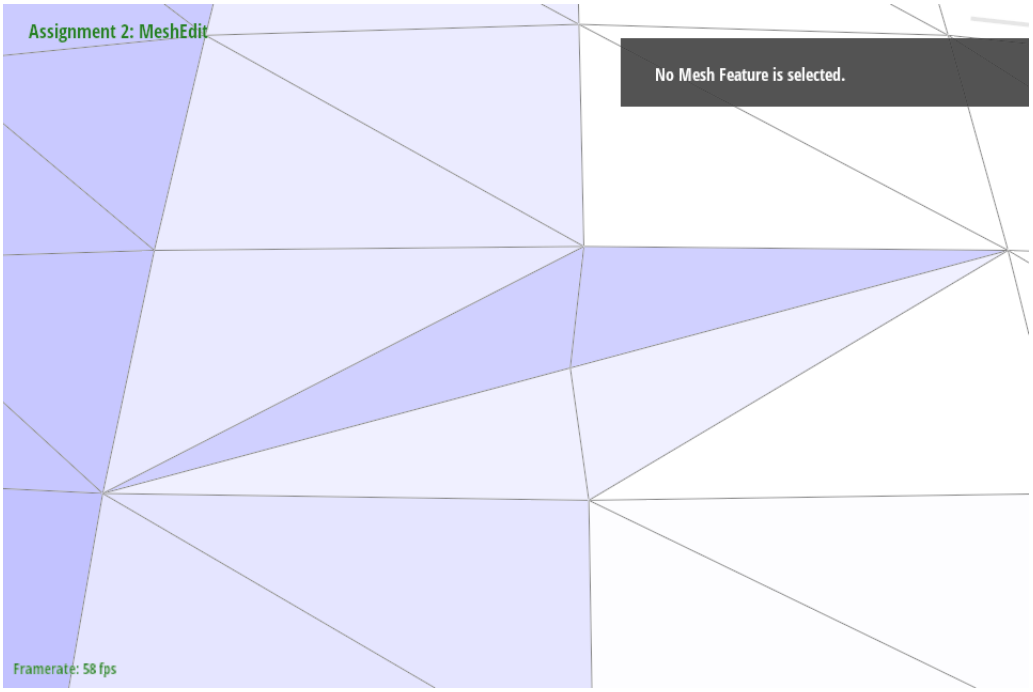

Assignment 2: MeshEdit
No Mesh Feature is selected.
Framerate: 57 fps


Assignment 2: MeshEdit
No Mesh Feature is selected.
Framerate: 59 fps

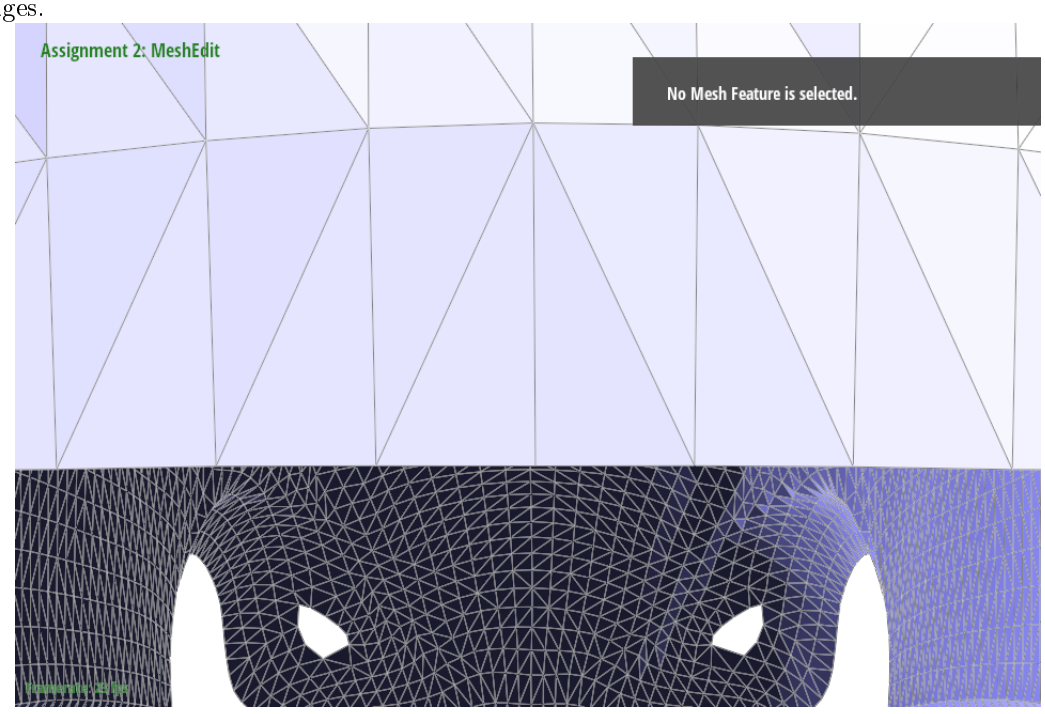


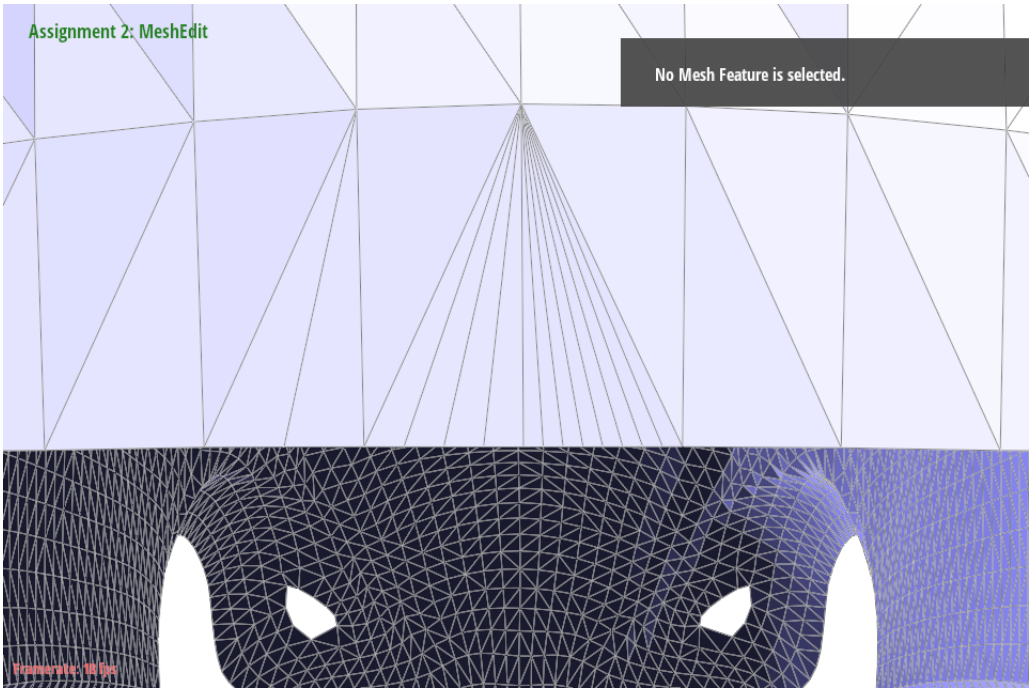
Screenshots of a mesh before and after a combination of both edge splits and edge flips:





Extra: screenshots of your implementation properly handling split operations on boundary edges.



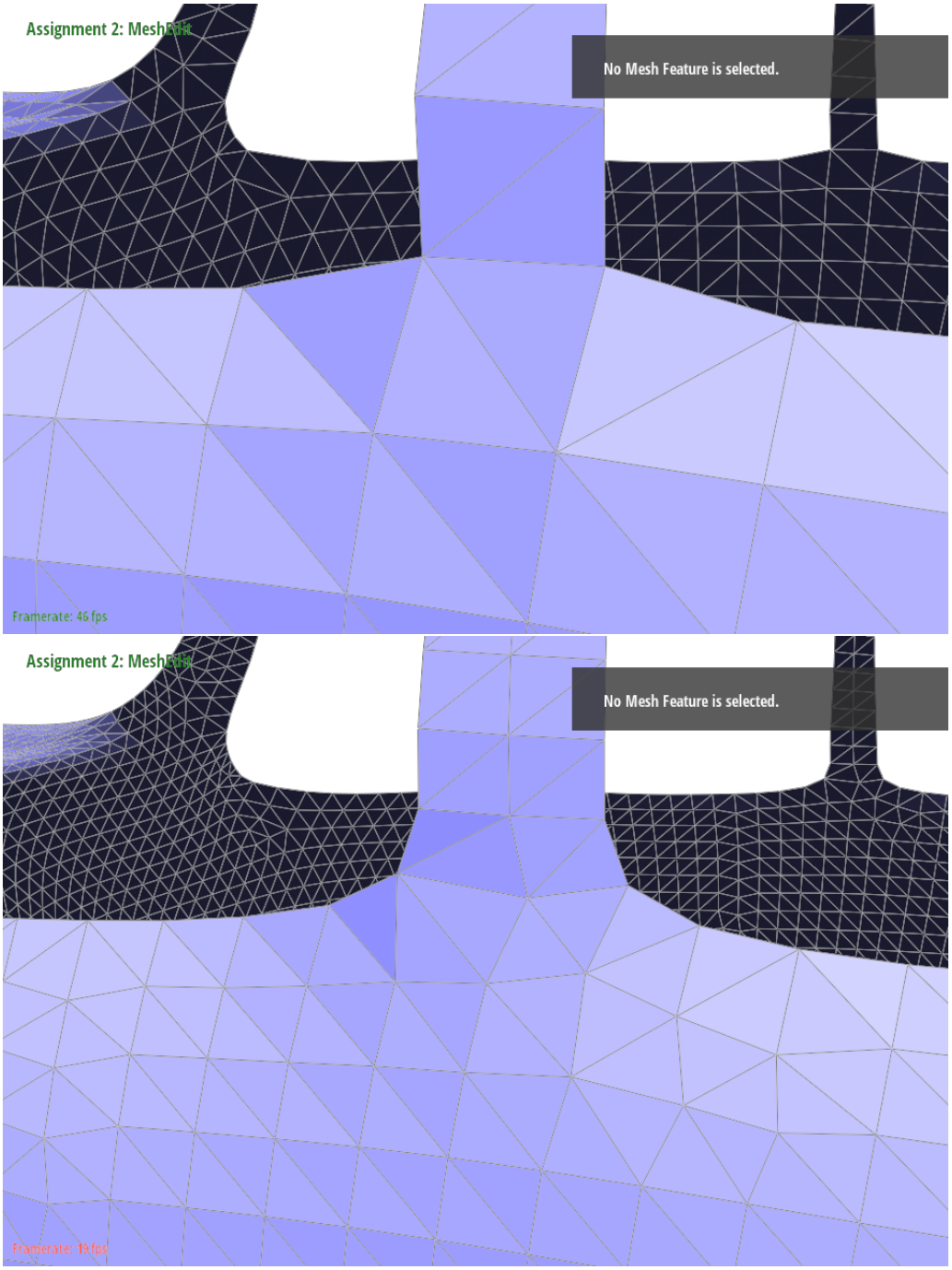


3.4 Loop subdivision for mesh upsampling

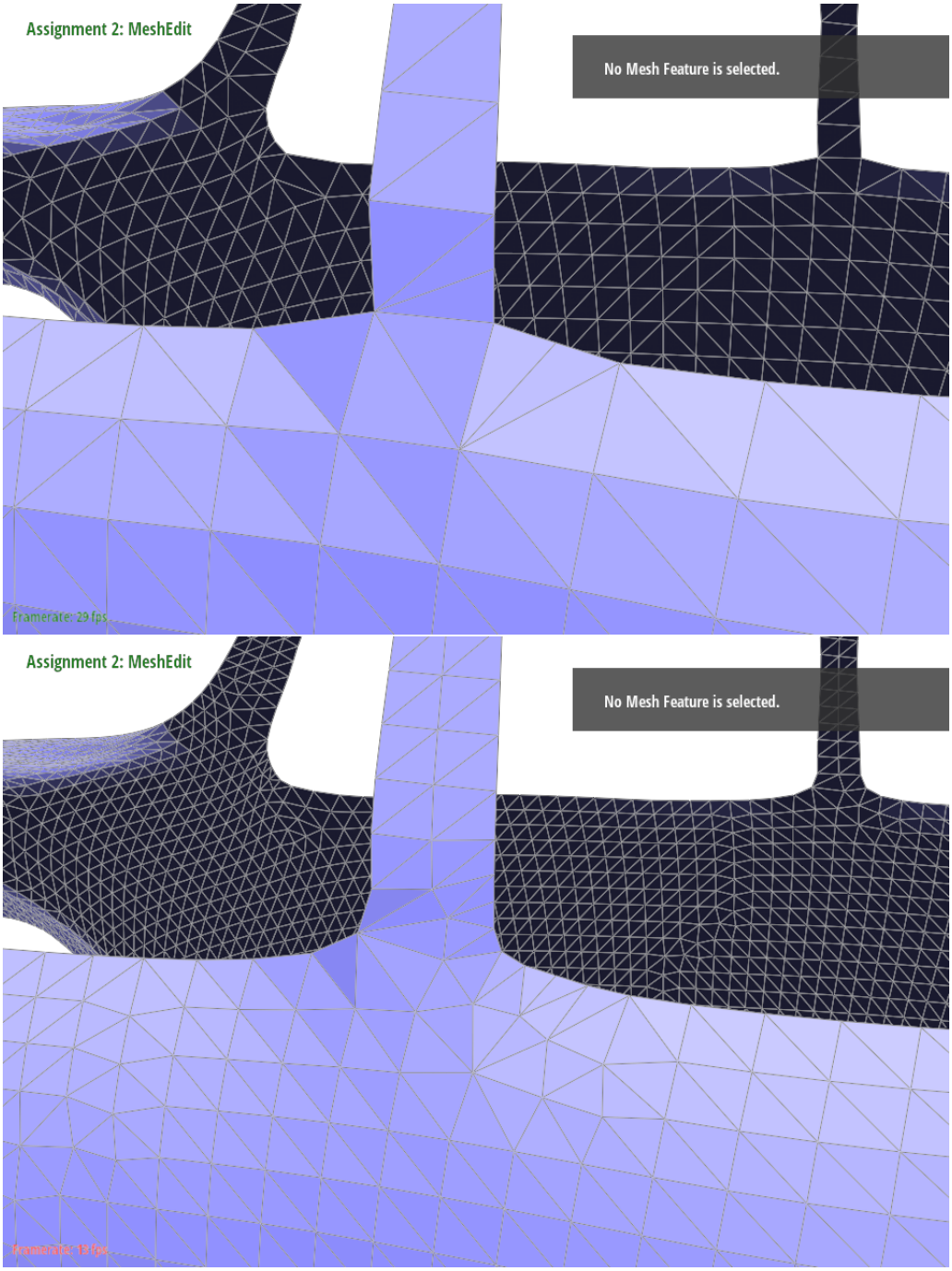
Another reason why 2D upsampling techniques do not easily translate to 3D versions is that if we simply increase the number of triangles by splitting each triangle, the shape of the mesh is not really changed.

In my implementation of loop subdivision, the new positions for all the vertices are computed and stored in the beginning. Then, the updated vertex positions associated with edges are computed and stored. Next, every edge in the mesh is splitted. Meanwhile, the position of each new vertex is updated. All the new edges that connects an old and new vertex are flipped. In the end, the position of each old vertex is updated.

Screenshots of how meshes behave after loop subdivision:



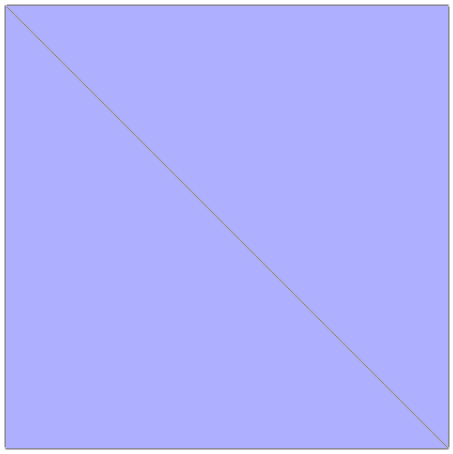
Sharp corners and edges become smoother. This effect can be reduced by pre-splitting the edges on the corner. In the following pictures, the edges on the right-hand-side corner are pre-splitted. After loop subdivision, the right-hand-side corner is shaper than the left-hand-side one.



Several iterations of loop subdivision on the cube:

Assignment 2: MeshEdit

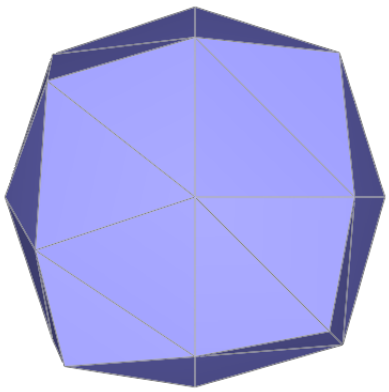
No Mesh Feature is selected.



Framerate: 137 fps

Assignment 2: MeshEdit

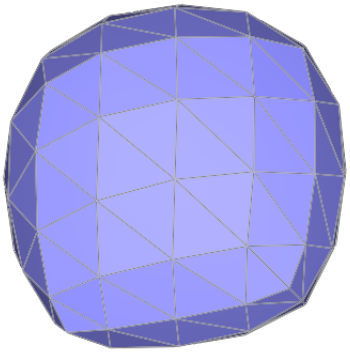
No Mesh Feature is selected.



Framerate: 181 fps

Assignment 2: MeshEdit

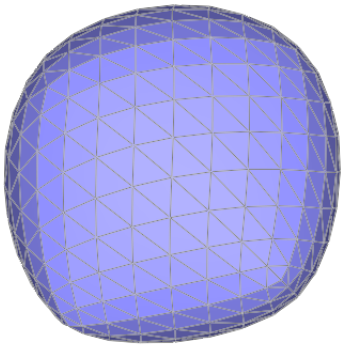
No Mesh Feature is selected.



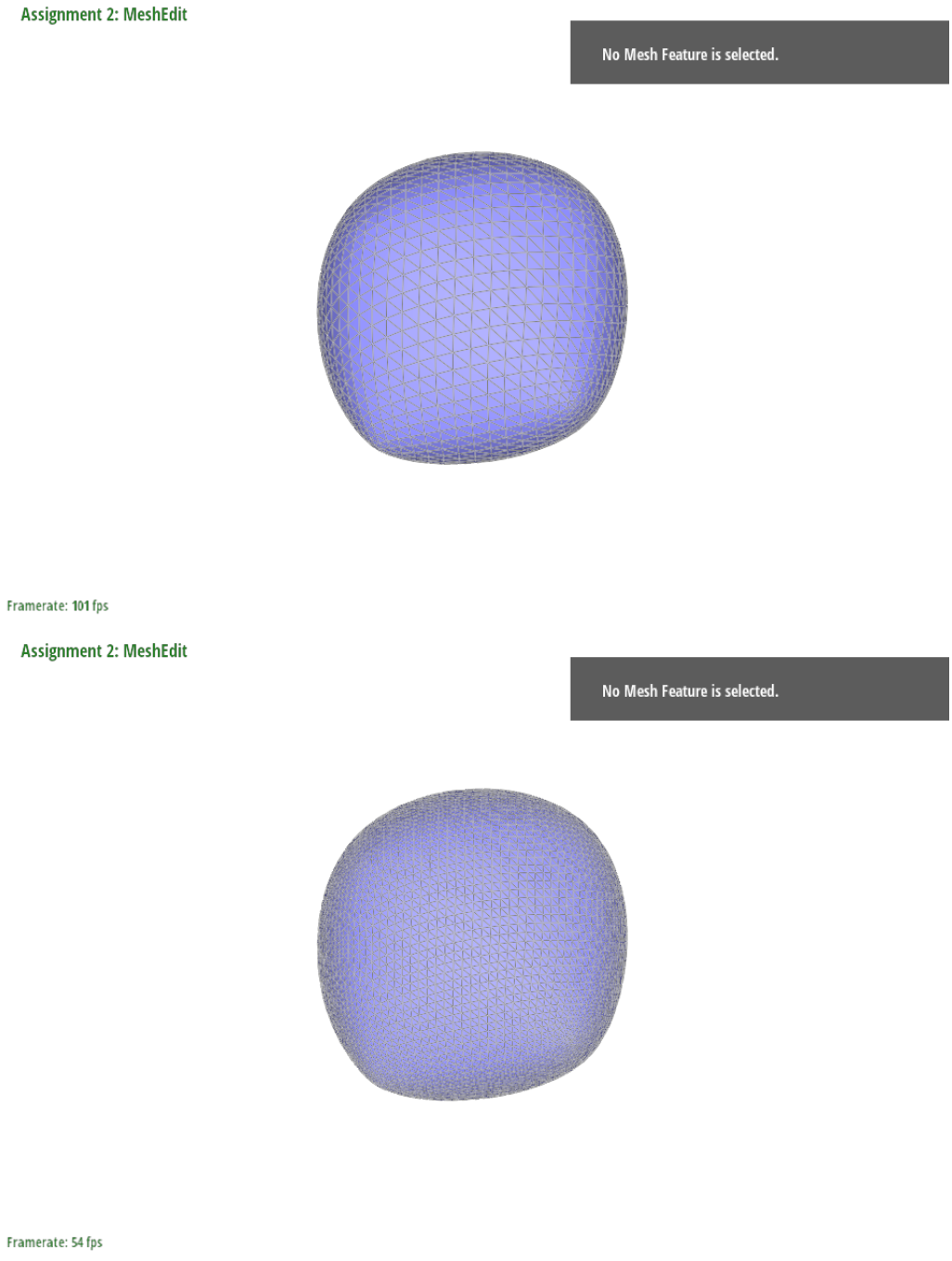
Framerate: 181 fps

Assignment 2: MeshEdit

No Mesh Feature is selected.



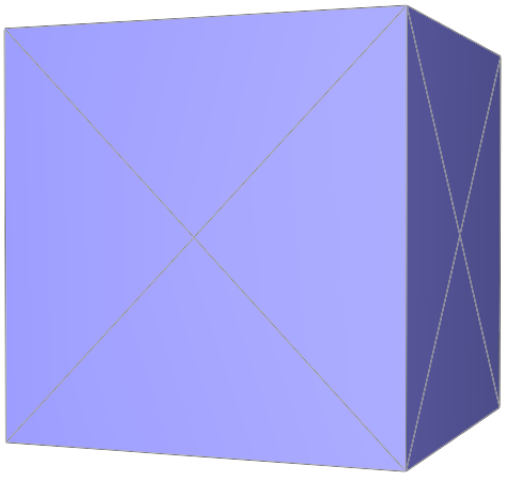
Framerate: 143 fps



In the above picture, the diagonal from the bottom-left to the upper-right is a little bit longer than the other one. The reason is that the original mesh is not really symmetric: on each surface of the cube, there is only one diagonal from the upper-left to the bottom-right. Therefore, I pre-process the cube so that the edges are symmetric in the beginning. Now, there are two diagonals on each surface of the cube. After loop subdivision, the resulting mesh becomes more symmetric.

Assignment 2: MeshEdit

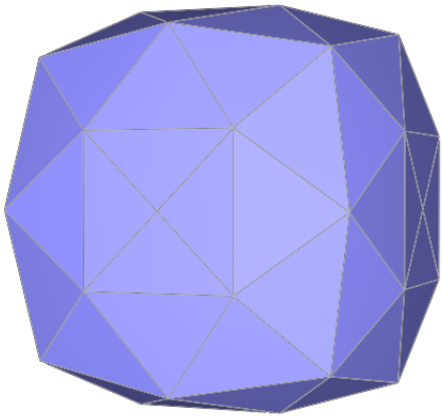
No Mesh Feature is selected.



Framerate: 91 fps

Assignment 2: MeshEdit

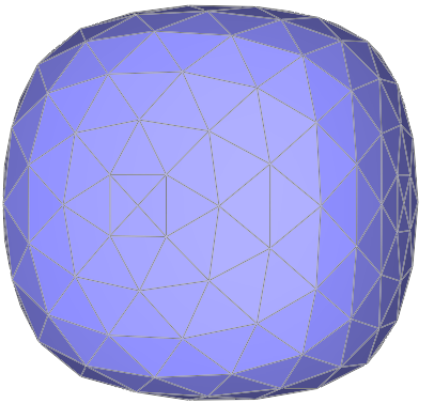
No Mesh Feature is selected.



Framerate: 84 fps

Assignment 2: MeshEdit

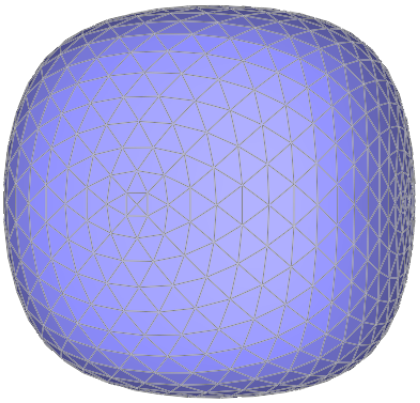
No Mesh Feature is selected.



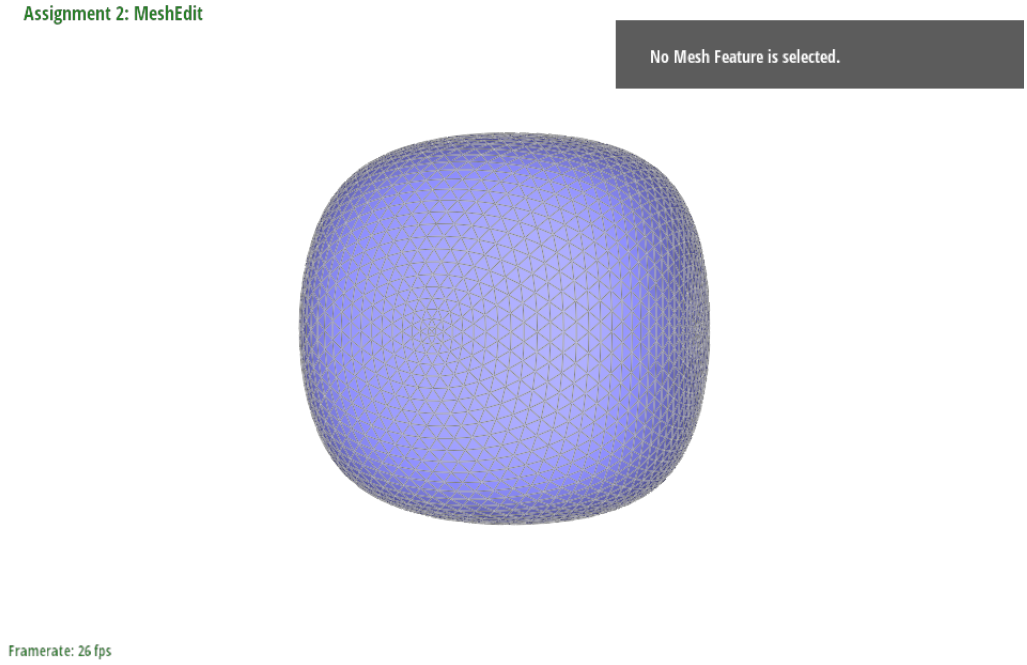
Framerate: 81 fps

Assignment 2: MeshEdit

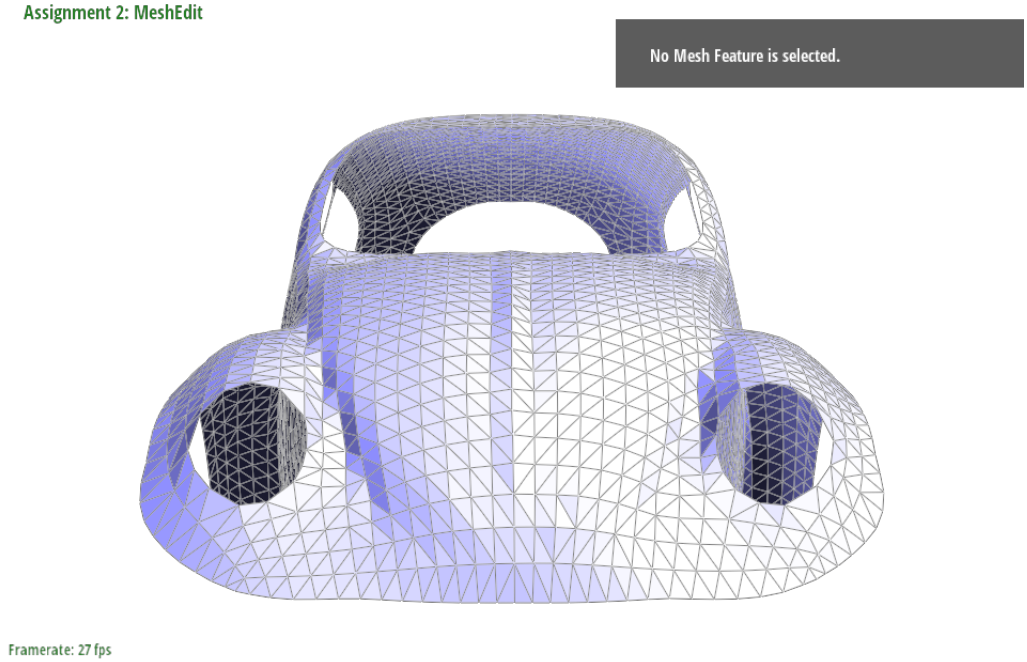
No Mesh Feature is selected.



Framerate: 55 fps

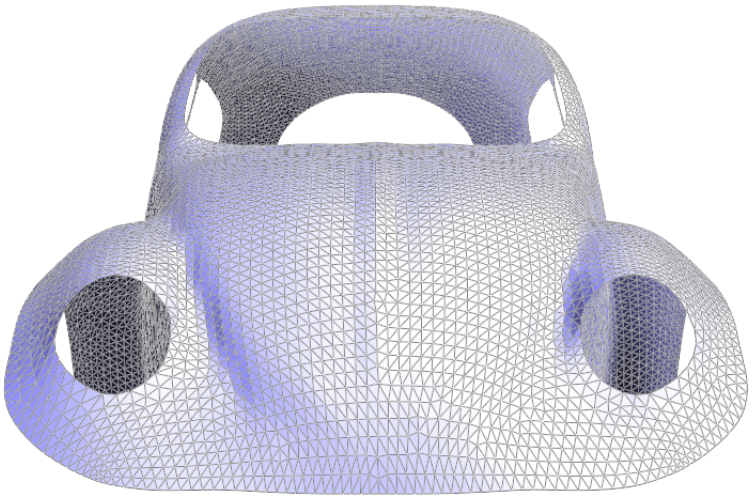


Extra: support of meshes with boundary



Assignment 2: MeshEdit

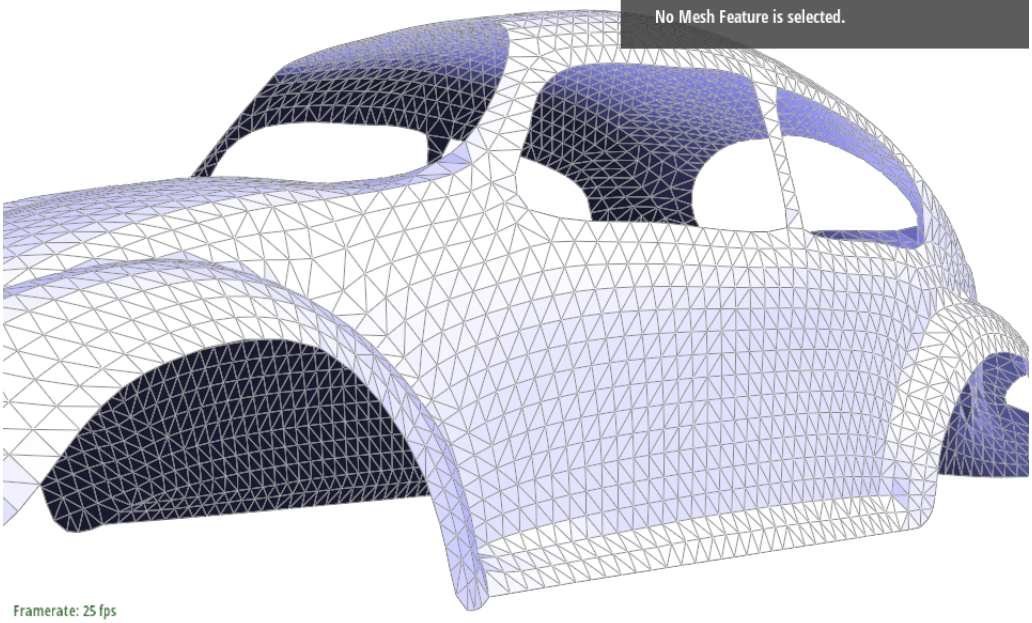
No Mesh Feature is selected.



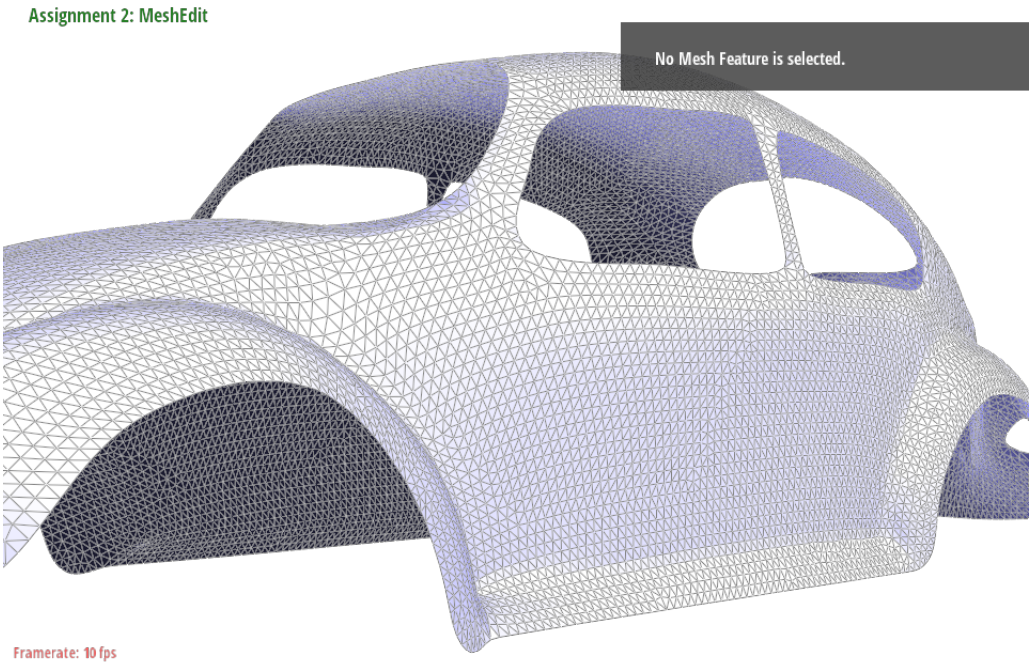
Framerate: 27 fps

Assignment 2: MeshEdit

No Mesh Feature is selected.



Framerate: 25 fps


Assignment 2: MeshEdit
No Mesh Feature is selected.
Framerate: 10 fps