ANUCG HW4 Report

u6740827 Heming Zhu

September 2023

1 Claim

This report and its related source code are Heming's intellectual work, built upon the provided skeleton ANU teaching crew provided.

2 1D de Casteljau Algorithm

To implement the 1D de Casteljsu algorithm, one has to repeatedly create new lines between the adjacent segments of the given line set, until there are no more adjacent segments. The method of creating new lines is first by doing linear interpolation on two adjacent segments, which gives two points for connecting that new line.

People tend to treat this algorithm stagable, where the first stage creates new segments from the given segments, then the second stage will create new segments from the product in stage 1, with the same method (i.e. connecting linear interpolation between adjacent segments). At the end of the algorithm, there will only be a stage that only has one segment to process, the linear interpolation on that point will be the target point on the Bezier Curve.

For a given line set with N vertexes, N stages will be created, where each of the stages contains a decreasing number of vertexes to process. The total number of vertexes needed to be processed can be calculated in the (1)

$$\#Segments = \sum_{i=1}^{N} i \tag{1}$$

In Heming's implementation, the data structure (named point_bucket) that stores all the vertexes is defined with the above-calculated size and will be stored or read from there. At the end of the algorithm, the target Bezier curve point will be collected from the final entry of the data structure. Please note that Heming didn't use C++ std::vector so that he could reach a better performance.

```
int stageNumber = control_points.size()-1;
int points_per_stage = 0;

int point_bucket_size = 0, bucket_idx=0;

for(int i=1; i<= stageNumber; i++){
    point_bucket_size += i;
}

//printf("bucket size %d\n", point_bucket_size);

Vector3f point_bucket[point_bucket_size];

for(int p_idx=0; p_idx < stageNumber; p_idx++){
    //linear interpolation
    point_bucket[bucket_idx] = control_points[p_idx]*(1-t) + control_points[p_idx+1]*t;
    bucket_idx++;
}

int bucket_idx_offset = bucket_idx;
int previous_idx_offset = 0;

for(int s_idx=1; s_idx<stageNumber; s_idx++){
    points_per_stage = stageNumber-s_idx;
    //printf("stage %d\n", s_idx);

    for(int p_idx=0; p_idx < points_per_stage; p_idx++){
        /*printf("read %d and %d in list to creat %d\n", bucket_idx-bucket_idx_offset + previous_idx_offset
                                                       , bucket_idx-bucket_idx_offset +previous_idx_offset +1
                                                       , bucket_idx);
        */
        point_bucket[bucket_idx] = (1-t) * point_bucket[bucket_idx-bucket_idx_offset + previous_idx_offset]
                                   + point_bucket[bucket_idx-bucket_idx_offset+previous_idx_offset+1] * t;
        bucket_idx++;
    }

    previous_idx_offset = bucket_idx_offset;
    bucket_idx_offset += points_per_stage;
}

return point_bucket[point_bucket_size-1];
```

Figure 1: Heming's code for 1d De Castlejai Algorithm

3 Bezier Surface

Two steps are required to create a Bezier surface out of a set of control points and a mesh.

3.1 Step 1: Create Control Lines

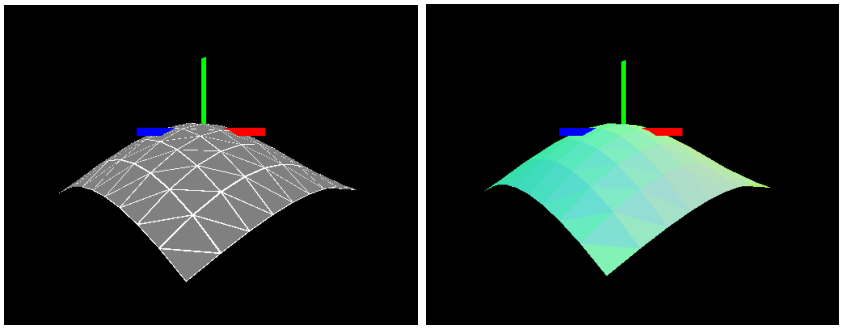
Assume the control points are stored in a M by N matrix. Then there will be M control lines, each drawn by the control points on the same column. If the mesh's resolution is R by R, then there will be R points on each control line.

3.2 Step 2: Scan Through the Line for a Surface

Now, if we form a matrix from the points of every control line, its dimension would be M by R. We can then create R control point sets, each with a size of M. With this, a number of R Bezier curves can be created, each with a resolution of R, which forms a Bezier surface.

3.3 Points To Mesh

Heming implemented a method to create triangles out of points, its source code is located at the end of the code that generates the Bezier patch.



(a) Wireframe (b) Normal

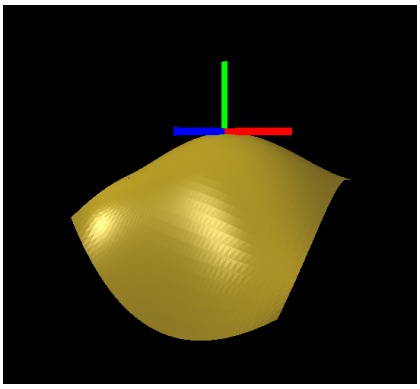


Figure 3: Phong shader, mesh twitched and rotated for a better view

4 Operator Manual of Heming's CAD

· Use the Left or Right Arrow Key to Rotate the mesh.
· Use the Up or Down Arrow Key to move the camera
forward or back from the mesh.
· Use Space Key to reset the view.
· Use "]" to select the next control point (the original selected control point is located at [0,0]), and use "[" to select the previous control point.
· Use "L" to descend the selected Control Point, and "U" to ascend the selected Control Point.
· Use "¡" and "¿" to change the resolution of mesh.

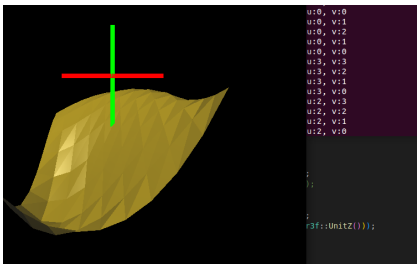


Figure 4: Morphed and Rotated Surface, the selected point is printed in the terminal

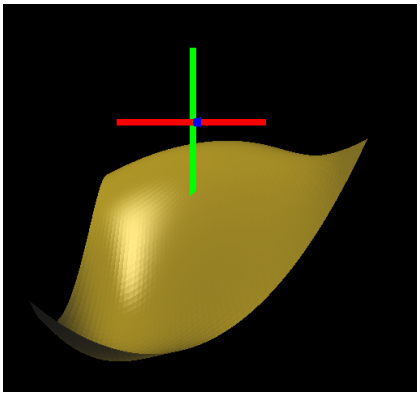
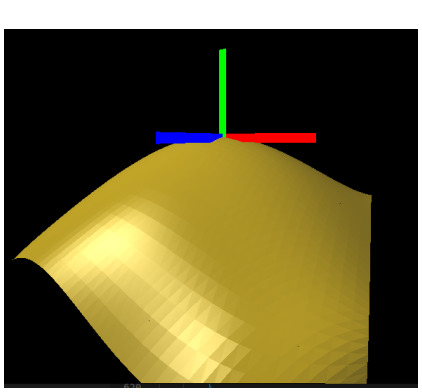
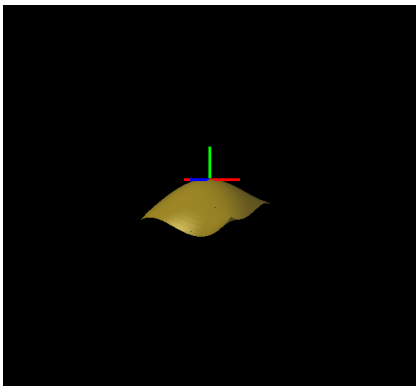


Figure 5: A higher resolution mesh



(a) Closer Look



(b) Further Look