Tutorial 9
CS3241 Computer Graphics (AY22/23)

November 7, 2023

Wong Pei Xian

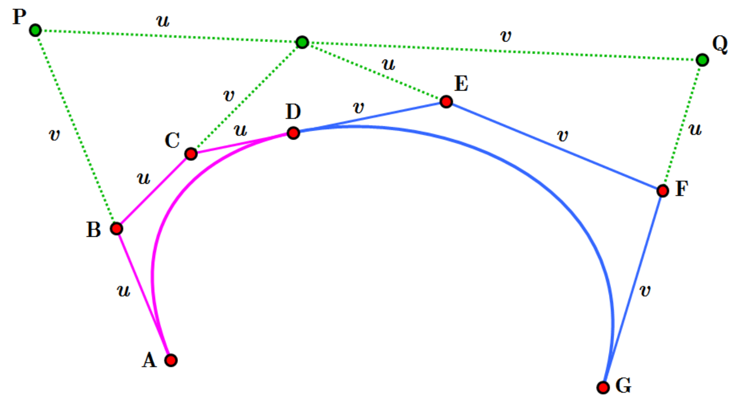
e0389023@u.nus.edu

Question 1

Considering the rendering efficiency and rendering quality on a standard polygon-based rasterization renderer, what are the disadvantages of representing surfaces using polygon mesh compared to using Bézier patches?

Adaptive subdivision

De Casteljau's algorithm can be used to **generate polygons** adaptively.
Less screen space ⇒ less levels of subdivision ⇒ less vertices.
More screen space ⇒ more levels of subdivision ⇒ more vertices.



Adaptive subdivision

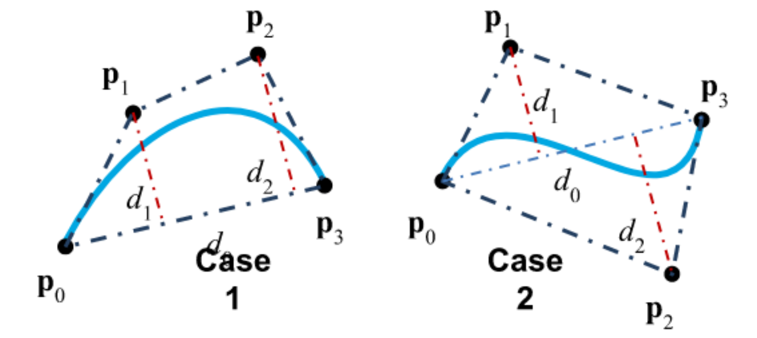
Compared to polygon mesh which has **fixed vertex/polygon count**, facing:

- poorer efficiency for small meshes which need less vertices
- poorer fidelity for large meshes which cannot achieve smooth curves

Question 2

Propose a way to measure the "flatness" of a cubic Bézier curve segment in 2D space. Can your method be extended to a cubic Bézier curve segment in 3D space?

Convex hull



Flatness estimate f

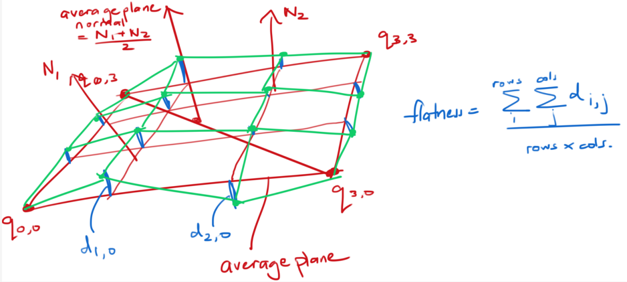
Case 1: $f = \max(d_1, d_2)$
Case 2: $f = \max(d_1, d_2)$

Q: How to determine cases 1 or 2?
A: determine if the intersection of the lines formed by p_1p_2 and p_0p_3 lies within p_0, p_3

Question 3

Propose a way to measure the “flatness” of a cubic Bézier **surface patch**.

Extending Q2



Define the "average plane" (in red) as $N \cdot p = N \cdot \frac{q_{0,0}+q_{0,3}+q_{3,0}+q_{3,3}}{4}$.
(where N is the average of the two normals of $q_{0,0}, q_{0,3}, q_{3,0}$ and $q_{3,0}, q_{0,3}, q_{3,3}$).

"Flatness" = average of distance between average plane and each point.

Question 4

Given 4 control points p_0, p_1, p_2, p_3 for a cubic Bézier curve segment $p(u)$, find the 4 control points q_0, q_1, q_2, q_3 for the cubic interpolating curve segment $q(u)$ such that $q(u) = p(u)$.

Cubic Bézier to Cubic interpolating

The corresponding $q_k = q(\frac{k}{3}) = p(\frac{k}{3})$ *(since* $q(u) = p(u)$*).*

$$
\begin{aligned}
q(0) &= \quad p_0 \\
q(\frac{1}{3}) &= \quad p(\tfrac{1}{3}) \\
q(\frac{2}{3}) &= \quad p(\tfrac{2}{3}) \\
q(1) &= \quad p_1
\end{aligned}
$$

Directly compute q from p.

Question 5

Given 4 control points q_0, q_1, q_2, q_3 for the cubic interpolating curve segment $q(u)$, find the 4 control points p_0, p_1, p_2, p_3 for a cubic Bézier curve segment $p(u)$ such that $p(u) = q(u)$.

Cubic interpolating to Cubic bezier

Let's look at our constraints:

$$
\begin{aligned}
p(0) &= p_0 \\
p(1) &= p_3 \\
p'(0) &= 3(p_1 - p_0) \\
p'(1) &= 3(p_3 - p_2)
\end{aligned}
$$

Cubic interpolating to Cubic bezier

Now we use the fact that $p(u) = q(u)$:

$$
\begin{aligned}
p_0 = p(0) = &\quad q(0) = q_0 \\
p_3 = p(1) = &\quad q(1) = q_3 \\
p'(0) = &\quad q'(0) = 3(p_1 - p_0) \\
p'(1) = &\quad q'(1) = 3(p_3 - p_2)
\end{aligned}
$$

For p_1, p_2 we obtain them via:

$$q'(0) = 3(p_1 - p_0) \Rightarrow \frac{q'(0)}{3} + p_0 = p_1$$

and

$$q'(1) = 3(p_3 - p_2) \Rightarrow p_3 - \frac{q'(1)}{3} = p_2$$

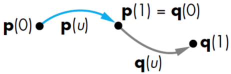
Question 6

Given two cubic Bezier curve segments, $p(u)$ and $q(u)$, that are to be joined together, where $p(1) = q(0)$. The control points of $p(u)$ are p_0, p_1, p_2, p_3 and the control points of $q(u)$ are q_0, q_1, q_2, q_3. How should the control points of $q(u)$ be positioned so that there is C^1 continuity at the join point of $p(u)$ and $q(u)$?

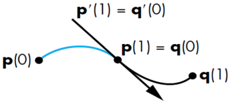
Continuity

Geometric and Parametric Continuity

- Consider two curve segments, $\mathbf{p}(u)$ and $\mathbf{q}(u)$
- If $\mathbf{p}(1) = \mathbf{q}(0)$, we say there is C^0 **parametric continuity** at the join point



- If $\mathbf{p}'(1) = \mathbf{q}'(0)$, we say there is C^1 **parametric continuity** at the join point



- If $\mathbf{p}'(1) = \alpha\mathbf{q}'(0)$, for some positive number α, we say there is G^1 **geometric continuity** at the join point
- We can extend the idea to higher derivatives and talk about C^n and G^n continuity

C^1 continuity

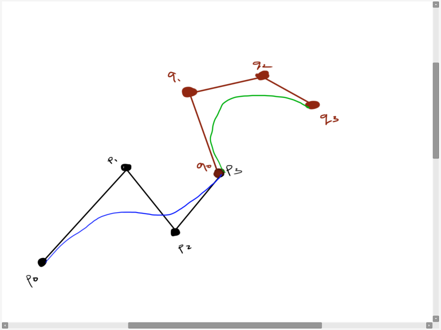
1. $p(1) = p(0)$
2. First derivative of p at end = first derivative of q at start: $p'(1) = q'(0)$.

$$
\begin{aligned}
p(1) &= p_3 = q_0 = q(0) \\
p'(1) &= 3(p_3 - p_2) = 3(q_1 - q_0) = q'(0)
\end{aligned}
$$

Question 7

Sketch two 2D curves that have C^0 continuity but not C^1 continuity.

Question 8

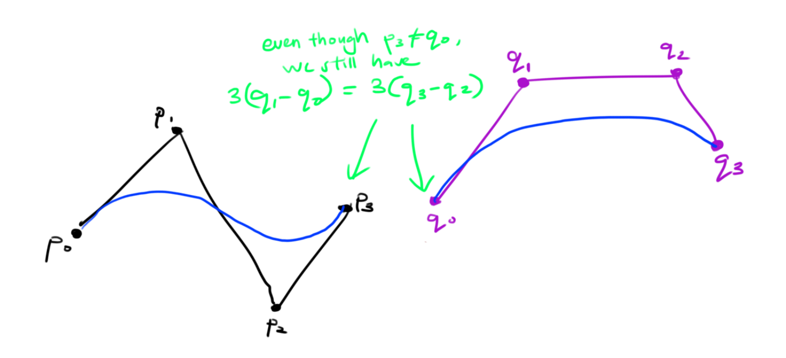


C^0 as both curves are joined $p_3 = q_0$.
Not C^1 as $p'(1) \neq q'(0)$.

Question 8

Sketch two 2D curves that have C^1 continuity but not C^0 continuity. Is it even possible to have such a situation?

Question 8



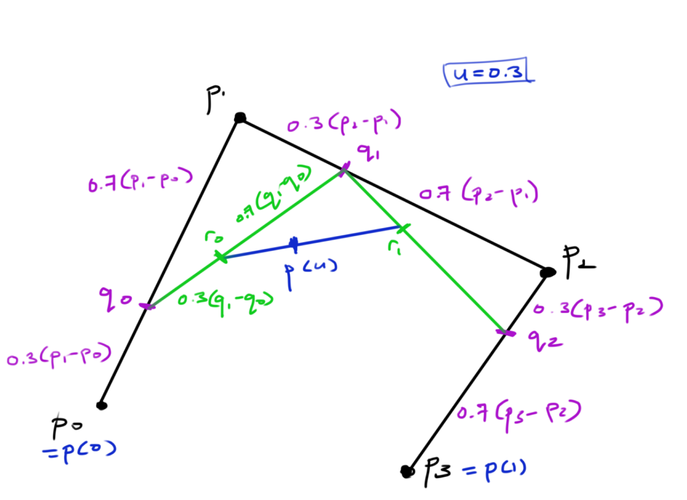
C^1 as $p'(1) = q'(0)$.
Not C^0 as $p(1) \neq q(0)$.

Question 9

Given 4 control points p_0, p_1, p_2, p_3 for a cubic Bézier curve segment $p(u)$, and any $0 \leq u \leq 1$ show that the De Casteljau algorithm produces the point $p(u)$.

$$p(u) = (1-u^{)}3p_0 + 3u(1-u)^2 p_1 + 3u^2(1-u)p_2 + u^3 p_3 \tag{1}$$

De Casteljau's Algorithm



Approach: Unpack the Lerps

Linear Interpolation of degree t along points a, b:

$$\mathrm{lerp}(t, a, b) = (1 - t)a + b \tag{2}$$

Unpacked all lerps

$$\mathbf{q}_0 = (1-u)\,\mathbf{p}_0 + u\,\mathbf{p}_1$$
$$\mathbf{q}_1 = (1-u)\,\mathbf{p}_1 + u\,\mathbf{p}_2$$
$$\mathbf{q}_2 = (1-u)\,\mathbf{p}_2 + u\,\mathbf{p}_3$$

$$\mathbf{r}_0 = (1-u)\,\mathbf{q}_0 + u\,\mathbf{q}_1$$
$$\mathbf{r}_1 = (1-u)\,\mathbf{q}_1 + u\,\mathbf{q}_2$$

$$\begin{aligned}
\mathbf{p} &= (1-u)\,\mathbf{r}_0 + u\,\mathbf{r}_1 \\
&= (1-u)\,((1-u)\,\mathbf{q}_0 + u\,\mathbf{q}_1) + u((1-u)\,\mathbf{q}_1 + u\,\mathbf{q}_2) \\
&= (1-u)^2\,\mathbf{q}_0 + 2u(1-u)\,\mathbf{q}_1 + u^2\,\mathbf{q}_2 \\
&= (1-u)^2\,((1-u)\,\mathbf{p}_0 + u\,\mathbf{p}_1) + 2u(1-u)\,((1-u)\,\mathbf{p}_1 + u\,\mathbf{p}_2) + u^2\,((1-u)\,\mathbf{p}_2 + u\,\mathbf{p}_3) \\
&= (1-u)^3\,\mathbf{p}_0 + 3u\,(1-u)^2\,\mathbf{p}_1 + 3u^2\,(1-u)\,\mathbf{p}_2 + u^3\,\mathbf{p}_3
\end{aligned}$$

Pseudocode

```
vec3 recursive_decasteljau(vector<vec3> points, float u) {
    // points.size() >= 1.
    if (points.size() == 1) {
        return points[0];
    }

    vector<vec3> interpolated_points(points.size() - 1);
    for (int i = 0; i < points.size() - 1; i++) {
        interpolated_points[i] = interpolate(points[i], points[i+1], u);
    }

    return recursive_decasteljau(interpolated_points, u);
}
```

Attendance taking

Thanks! Get the slides here after the tutorial.

https://trxe.github.io/cs3241-notes