Rasterization, Splines, and Curves 4

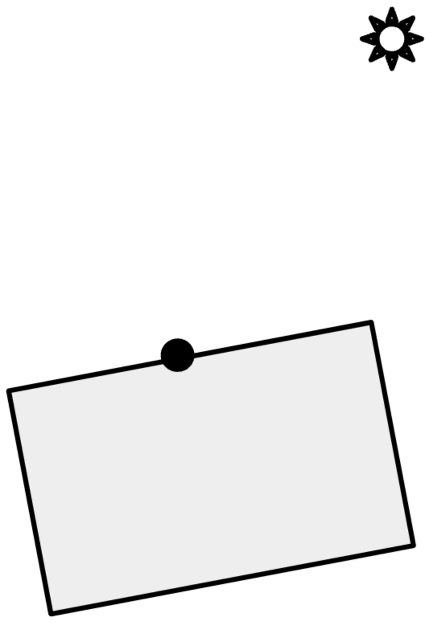
CS 184: Foundations of Computer Graphics

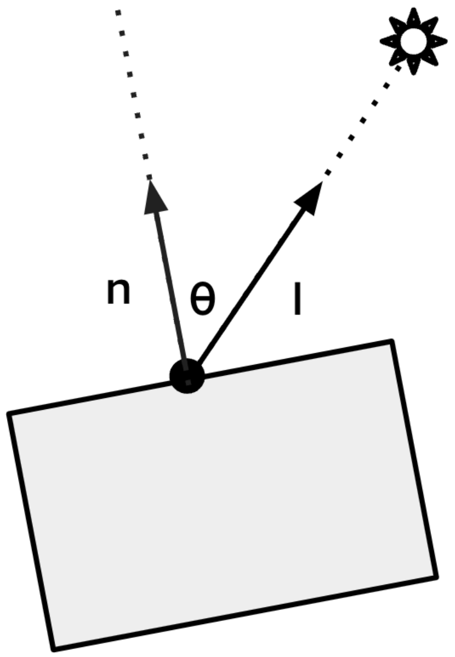
1 Graphics Pipeline — Lightning Round!

1. Name and describe the three terms in the Blinn-Phong Reflection Model.

Solution: Ambient, Diffuse, Specular lighting.

2. A light source shines on a tilted surface. Draw the light direction vector, **l**, and the normal vector, **n**, at the given point.





Solution:

3. What is the light per unit area on this surface proportional to, according to Lambert's cosine law?

Solution: $\mathbf{l} \cdot \mathbf{n} = \cos\theta$. Light intensity reflected off surface is proportional to cosine of the angle between the light direction vector and surface normal vector.

4. Complete the following implementation of the Z-Buffer Algorithm in C++.

```
const int WIDTH = 800; // Width of framebuffer
const int HEIGHT = 600; // Height of framebuffer

struct Color {
    float r, g, b;
};

struct Sample {
    int x, y;
    float z;
    Color color;
};

struct Triangle {
    std::vector<Sample> samples;
};

void zBufferAlgorithm(const _______________________________________ triangles,
                            _______________________________________ framebuffer,
                            _______________________________________ zbuffer) {
```

```
    for (const Triangle& T : triangles) {
        for (const Sample& sample : T.samples) {
            int x = sample.x;
            int y = sample.y;
            float z = sample.z;

            if (x >= 0 && x < WIDTH && y >= 0 && y < HEIGHT) {
                if (_______________________________________) {
                    framebuffer[x][y] = sample.color;
                    _______________________________________;
                }
            }
        }
    }
}
```

Solution:

This solution uses **std::vector**, but other C++ implementations of lists could also work.

- `std::vector<Triangle>&`
- `std::vector<std::vector<Color>>&`
- `std::vector<std::vector<float>>&`
- `z < zbuffer[x][y]`
- `zbuffer[x][y] = z`

5. Prior to running this algorithm, what should the Z-buffer values be initialized to?

Solution:

`std::numeric_limits<float>::infinity()`

(Infinity.)

2 A Polynomial Interpolation

Our goal is to fit a polynomial to given points and derivatives of a curve. We can solve this problem by formulating it as a system of linear equations in the coefficients of the polynomial.

1. List all degree 2 polynomials satisfying: $f(0) = 1$, $f(1) = 2$, $f(2) = 5$.

Solution: Let $f(t) = at^2 + bt + c$ be the degree 2 polynomial. The system of constraints we get is:

$$\begin{aligned} c &= 1 \\ a + b + c &= 2 \\ 4a + 2b + c &= 5 \end{aligned}$$

Solving the system yields a unique solution $a = 1, b = 0, c = 1$, so $f(t) = t^2 + 1$ is the only degree 2 polynomial.

2. How many degree 3 polynomials satisfy the given constraints?

Solution: There are infinitely many such polynomials. Intuitively, three constraints on four coefficients (since a degree 3 polynomial has four coefficients) leave one free parameter, so there must be infinitely many degree 3 solutions.

Detailed explanation (optional). We know from the first question that $f(t) = t^2 + 1$ is the unique degree 2 polynomial satisfying

$$f(0) = 1, \quad f(1) = 2, \quad f(2) = 5.$$

Now let $f(t)$ be a degree 3 polynomial. We can write

$$f(t) \;=\; \underbrace{(t^2+1)}_{f_1(t)} + \underbrace{(at^3 + bt^2 + ct + d)}_{f_2(t)},$$

where $f_1(t)$ already satisfies the constraints. Therefore, the polynomial

$$f_2(t) := at^3 + bt^2 + ct + d$$

must vanish at $t = 0, 1, 2$:

$$f_2(0) = 0, \quad f_2(1) = 0, \quad f_2(2) = 0.$$

From $f_2(0) = 0$, we get $d = 0$. From $f_2(1) = 0$, $a + b + c = 0$. From $f_2(2) = 0$, $8a + 4b + 2c = 0$.

To solve this system, subtract $\left(\frac{1}{2}\right) f_2(2)$ from $f_2(1)$

$$(4a + 2b + c) \;-\; (a + b + c) = 0 \quad\Longrightarrow\quad 3a + b = 0 \quad\Longrightarrow\quad b = -3a.$$

Next, substitute $b = -3a$ into $a + b + c = 0$:

$$a - 3a + c = 0 \quad\Longrightarrow\quad -2a + c = 0 \quad\Longrightarrow\quad c = 2a.$$

Hence,

$$f_2(t) = a\,(t^3 - 3t^2 + 2t).$$

Therefore, *every* degree-3 polynomial satisfying the original constraints can be written as

$$f(t) = t^2 + 1 + a\,(t^3 - 3t^2 + 2t),$$

for any real a. Since a is a free parameter that can take any real value, there are infinitely many solutions.

3. Suppose we have a list of constraints:

$$f(0) = p_0,\ f'(0) = d_0,\ f(1) = p_1,\ f'(1) = d_1,\ \ldots,\ f(k) = p_k,\ f'(k) = d_k.$$

Since we have $2(k+1)$ constraints (one function and one derivative condition per point), the unique interpolating polynomial must have degree $2k+1$.

For a function f, what are the tradeoffs when either

- solving for a single degree $2k+1$ polynomial, versus
- taking the point and derivative constraints at i and $i-1$ for $i = 1, \ldots, k$ and using them to fit k cubic Hermite splines?

Solution: If we solve for a single high-degree polynomial, it will have infinitely many continuous derivatives. However, it may exhibit unpredictable behavior between the control points, and furthermore, changing a single constraint will affect the entire curve.

If we solve for k cubic Hermite splines, then the resulting curve will be continuous and have a continuous first derivative. The curve will have infinitely many continuous derivatives within each segment but may have a discontinuous second derivative at the control points. However, changing a single constraint will only affect two of the cubic splines: those that have the control point as an endpoint.

4. Consider a cubic polynomial $f(t) = at^3 + bt^2 + ct + d$ that satisfies the following conditions:

$$\begin{aligned} f(0) &= f_0, \\ f(1) &= f_1, \\ f''(0) &= f_0'', \\ f''(1) &= f_1''. \end{aligned}$$

Write the matrix that, when inverted and applied to the vector $(f_0, f_1, f_0'', f_1'')^T$, allows you to recover the coefficients a, b, c, and d of the polynomial.

Solution:

$$\begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 2 & 0 & 0 \\ 6 & 2 & 0 & 0 \end{pmatrix}^{-1} \begin{pmatrix} f_0 \\ f_1 \\ f_0'' \\ f_1'' \end{pmatrix}$$

5. Given the numerical inverse of the matrix:

$$\begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} 0 & 0 & -1/6 & 1/6 \\ 0 & 0 & 1/2 & 0 \\ -1 & 1 & -1/3 & -1/6 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ f_0'' \\ f_1'' \end{pmatrix}$$

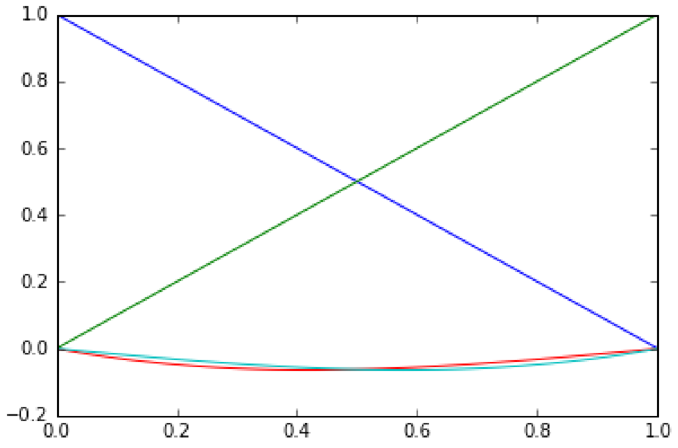
what are the **basis polynomials** that allow expressing $f(t)$ in terms of f_0, f_1, f_0'',, f_1''?

Solution:

The basis polynomials for this problem are:

$$G_0(t) = -t + 1, \quad G_1(t) = t, \quad G_2(t) = -\frac{t^3}{6} + \frac{t^2}{2} - \frac{t}{3}, \quad G_3(t) = \frac{t^3}{6} - \frac{t}{6}.$$

Here's a plot of these functions:



These basis polynomials form a set of functions that allow us to express any cubic polynomial satisfying the given constraints. The interpolating function is given by:

$$f(t) = f_0 G_0(t) + f_1 G_1(t) + f_0'' G_2(t) + f_1'' G_3(t).$$

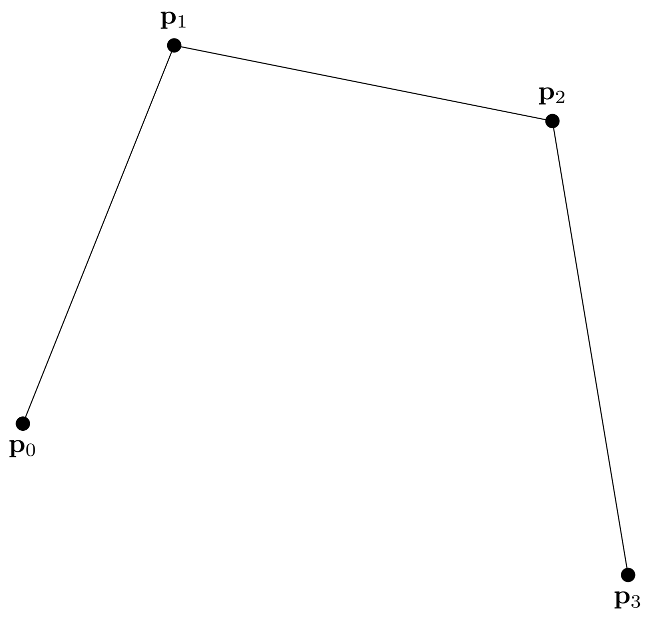
$f(t)$ matches the given function values and second derivatives at $t = 0$ and $t = 1$.

3 de Casteljau's algorithm

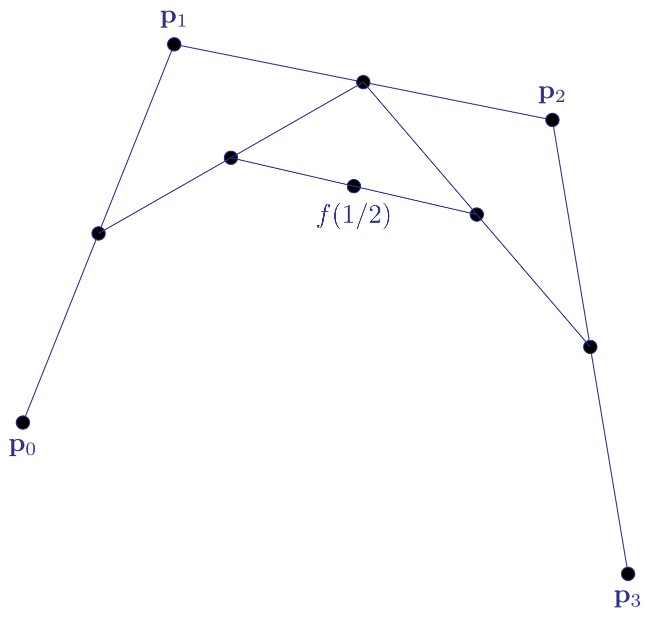
Given $k+1$ points $\mathbf{p}_0, \ldots, \mathbf{p_k}$, create a new set of k points $\mathbf{p'_0}, \ldots, \mathbf{p'_{k-1}}$ by computing $\mathbf{p'_i} = \text{lerp}(\mathbf{p_i}, \mathbf{p_{i+1}}, t)$, where $\text{lerp}(\mathbf{p_i}, \mathbf{p_{i+1}}, t) = (1-t)\mathbf{p_i} + t\mathbf{p_{i+1}}$.

Perform k times to yield a single point, $f(t)$.

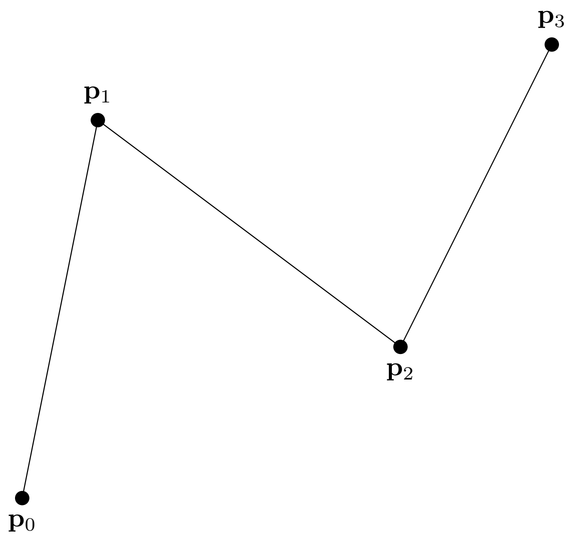
1. Use de Casteljau's algorithm to construct $f(1/2)$ with the given control points.



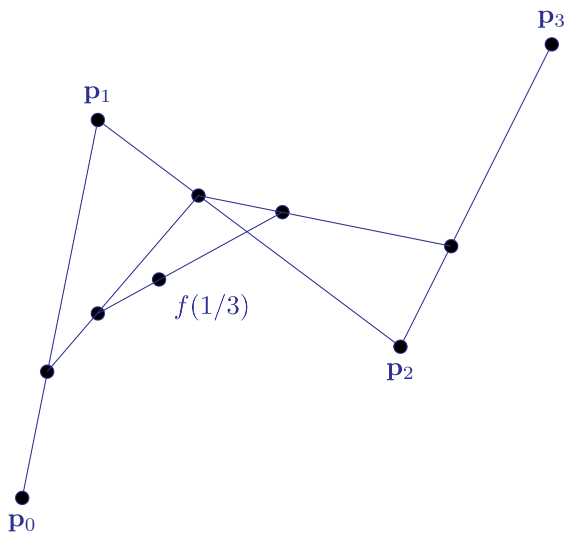
Solution:



2. Use de Casteljau's algorithm to construct $f(1/3)$ with the given control points.



Solution:



3. Show that the point with parameter t on the Bézier curve with control points $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ is given by $s^3\mathbf{p}_0 + 3s^2t\mathbf{p}_1 + 3st^2\mathbf{p}_2 + t^3\mathbf{p}_3$, where $s = 1 - t$. (Hint: apply de Casteljau's algorithm algebraically to the control points. With this setup, linear interpolation between two points $\mathbf{q_0}$ and $\mathbf{q_1}$ looks like $s\mathbf{q_0} + t\mathbf{q_1}$.)

Solution:

Level 0: $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$

Level 1: $\mathbf{b}_0^1(t) = s\mathbf{p}_0 + t\mathbf{p}_1$
$\mathbf{b}_1^1(t) = s\mathbf{p}_1 + t\mathbf{p}_2$
$\mathbf{b}_2^1(t) = s\mathbf{p}_2 + t\mathbf{p}_3$

$$\begin{aligned}
\text{Level 2:}\quad \mathbf{b}_0^2(t) &= s(s\mathbf{p}_0 + t\mathbf{p}_1) + t(s\mathbf{p}_1 + t\mathbf{p}_2) = s^2\mathbf{p}_0 + 2st\mathbf{p}_1 + t^2\mathbf{p}_2 \\
\mathbf{b}_1^2(t) &= s(s\mathbf{p}_1 + t\mathbf{p}_2) + t(s\mathbf{p}_2 + t\mathbf{p}_3) = s^2\mathbf{p}_1 + 2st\mathbf{p}_2 + t^2\mathbf{p}_3 \\
\text{Level 3:}\quad \mathbf{b}_0^3(t) &= s(s^2\mathbf{p}_0 + 2st\mathbf{p}_1 + t^2\mathbf{p}_2) + t(s^2\mathbf{p}_1 + 2st\mathbf{p}_2 + t^2\mathbf{p}_3) \\
&= s^3\mathbf{p}_0 + 3s^2t\mathbf{p}_1 + 3st^2\mathbf{p}_2 + t^3\mathbf{p}_3
\end{aligned}$$

4. What is this matrix product? (Hint: *don't* expand it. Instead, think about what each matrix in the product does. How are they related to de Casteljau's algorithm?)

$$\begin{pmatrix} s & t \end{pmatrix} \begin{pmatrix} s & t & 0 \\ 0 & s & t \end{pmatrix} \begin{pmatrix} s & t & 0 & 0 \\ 0 & s & t & 0 \\ 0 & 0 & s & t \end{pmatrix}$$

Solution: Without doing any additional computation, the matrix product must be $\begin{pmatrix} s^3 & 3s^2t & 3st^2 & t^3 \end{pmatrix}$.

The rest of the solution assumes that $s = 1 - t$, but the same logic applies to any s (in Question 3, we made no assumptions on s). Let's consider the result of applying this matrix product to a set of points,

$$\begin{pmatrix} \mathbf{p}_0^T \\ \mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \mathbf{p}_3^T \end{pmatrix}.$$

Starting from the right, the first matrix maps this input to the three points that result in applying one iteration of de Casteljau's algorithm,

$$\begin{pmatrix} s\mathbf{p}_0^T + t\mathbf{p}_1^T \\ s\mathbf{p}_1^T + t\mathbf{p}_2^T \\ s\mathbf{p}_2^T + t\mathbf{p}_3^T \end{pmatrix} = \begin{pmatrix} \mathbf{b}_0^1(s,t)^T \\ \mathbf{b}_1^1(s,t)^T \\ \mathbf{b}_2^1(s,t)^T \end{pmatrix}.$$

The other two matrices perform the remaining 2 iterations of the algorithm. As we saw in Question 3, applying this matrix product is equivalent to the map

$$\begin{pmatrix} \mathbf{p}_0^T \\ \mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \mathbf{p}_3^T \end{pmatrix} \mapsto s^3\mathbf{p}_0^T + 3s^2t\mathbf{p}_1^T + 3st^2\mathbf{p}_2^T + t^3\mathbf{p}_3^T.$$

Hence, the matrix product must be $\begin{pmatrix} s^3 & 3s^2t & 3st^2 & t^3 \end{pmatrix}$.

Alternatively, you could just do the computation... but why would you do that?

$$\begin{aligned}
&\begin{pmatrix} s & t \end{pmatrix} \begin{pmatrix} s & t & 0 \\ 0 & s & t \end{pmatrix} \begin{pmatrix} s & t & 0 & 0 \\ 0 & s & t & 0 \\ 0 & 0 & s & t \end{pmatrix} \\
&= \begin{pmatrix} s^2 & 2st & t^2 \end{pmatrix} \begin{pmatrix} s & t & 0 & 0 \\ 0 & s & t & 0 \\ 0 & 0 & s & t \end{pmatrix} \\
&= \begin{pmatrix} s^3 & 3s^2t & 3st^2 & t^3 \end{pmatrix}
\end{aligned}$$

5. For a Bézier curve defined by 3 control points, what is the degree of the polynomial that results from de Casteljau's algorithm? What about for k points?

Solution: Three points yields a quadratic polynomial, and k points yields a polynomial of degree $k - 1$.[1]

In Question 3, we saw that

$$\mathbf{b}_0^3(t) = \binom{3}{0}(1-t)^3\mathbf{p}_0 + \binom{3}{1}(1-t)^2 t\mathbf{p}_1 + \binom{3}{2}(1-t)t^2\mathbf{p}_2 + \binom{3}{3}t^3\mathbf{p}_3.$$

More generally, we can define Bézier curves in Bernstein form, i.e.

$$\mathbf{b}_0^n(t) = \sum_{j=0}^{n} \mathbf{p}_j \binom{n}{i}(1-t)^{n-i}t^i.$$

This is a polynomial with degree n. If we have 3 points, then $n = 2$, and the polynomial is quadratic in t. If we have k points, then $n = k - 1$, and the polynomial has degree k.

[1]Here, we consider the control points $\mathbf{p}_j$ to be variables and the resulting function to be a polynomial in t. If you can fix $\mathbf{p}_j$'s, it is possible to obtain a polynomial with lower degree. For example, if every $\mathbf{p}_j = \mathbf{0}$, then the resulting polynomial is $\mathbf{0}$.