

Week 1 - Workshop

1. *Why curves?*

Curves are needed in order to approximate real-life or intended behaviour better and/or make it visually more accurate and appealing. Curves can be used for modelling realistic objects or even for things that are not necessarily visible like camera paths, character motions or other complex trajectories. If you think of early games, characters would only move right, left, up and down with was causing jittery motion.

2. *Then, why there are more types of curves?*

As with any problem, there are multiple ways to 'solve' a curve. Some offer more accuracy over what you need to model, some are more scalable and do not require solving high degree polynomials, but require different data structures and are more robust.

E.g. Whilst you can technically draw a curve by using the equation of a circle and drawing arcs or similarly polynomial curves, is that really useful for visual artists that want to create curved surfaces? They would always have to find the parameters for the equation and so on.

3. *Who really first invented Bezier curves and what for?*

Similar to the Newton and Leibniz situation, one often gets more credit for coming up with the same thing as a previous scientist.

The curves were first developed in 1959 by [Paul de Casteljau](#) using [de Casteljau's algorithm](#). This was a bit earlier than Bezier but could not be published because of industrial privacy issues. De Casteljau worked at Citroen and Bezier at Renault (both French car manufacturers). Important to note that France was a scientific powerhouse in the 19th and 20th century.

The curves were developed in order to generate curved surfaces in Computer Aided Design (CAD).

Fun fact: ACM SIGGRAPH (biggest graphics association in the world affiliated with the biggest graphics conference) offered Bezier an award for life-time contributions to Computer Graphics.

4. *In the following figure, Which pair of curves is Bezier?*

Hint: Look for the one that starts and ends on the control points.

5. *What are the pros and cons of Bezier vs B-Spline curve?*

Bezier: Very flexible and computationally cheap to generate. They are not robust to changes, changing one control point can drastically change the entire shape.

B-Spline: Less flexible, essentially a chain of repeated curves (using one of the curve equations) that are “glued” together. Because it is segmented, only neighbouring knots will be affected.

6. *True or false: In Bezier, the degree of the polynomial defining the curve segment is one less than the number of defining polygon points*

Answer: Very true

7. *True or false: B-Spline allows the order of the basis function so the degree of the resulting curve is independent of the number of vertices.*

Answer: True --- Again, the degree of the B-spline is an input describing the basis function, independent of the number of vertices. In Bezier, the degree is given by the control points

8. *What is the general function $f(t)$ for quadratic and cubic Bezier curves? Use these to derive the 4x4 matrix so it no longer looks scary.*

The key to understanding this question is to understand how Linear interpolation looks graphically.

In the linear Bezier curve:

$$\mathbf{B}(t) = \mathbf{P}_0 + t(\mathbf{P}_1 - \mathbf{P}_0)$$

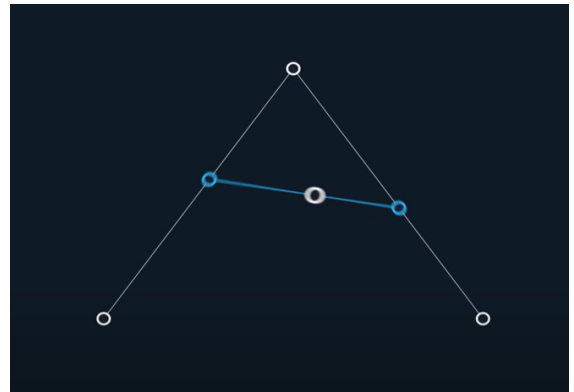
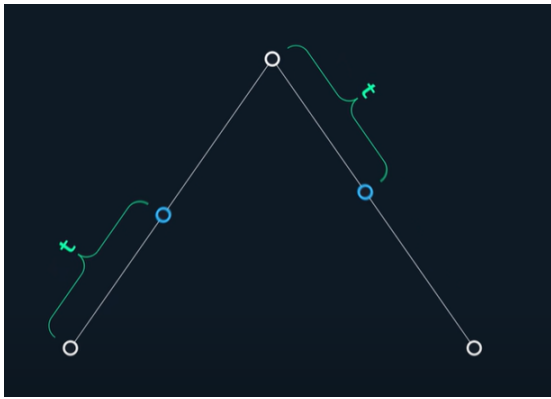
As t varies from 0 to 1, we essentially see a straight line being drawn

Quadratic Bezier Curve:

When we add a new point we basically commit a cascade of lerps.

We'll have a lerp between P_0 and P_1 and a lerp between P_1 and P_2 . The lerp results are the blue dots and value t . If we lerp in between those 2 points we'll get a Point P describing the curve we are looking for and the general equation is this.

$$\mathbf{B}(t) = (1 - t)^2 \mathbf{P}_0 + 2(1 - t)t \mathbf{P}_1 + t^2 \mathbf{P}_2, \quad 0 \leq t \leq 1.$$



Cubic Bezier Curves

We just add another point and we will interpolate in between 4 points. This means

$$\mathbf{B}(t) = (1 - t)^3 \mathbf{P}_0 + 3(1 - t)^2 t \mathbf{P}_1 + 3(1 - t) t^2 \mathbf{P}_2 + t^3 \mathbf{P}_3, \quad 0 \leq t \leq 1.$$

lerp(P0,P1,t) -> the resulting point will be A

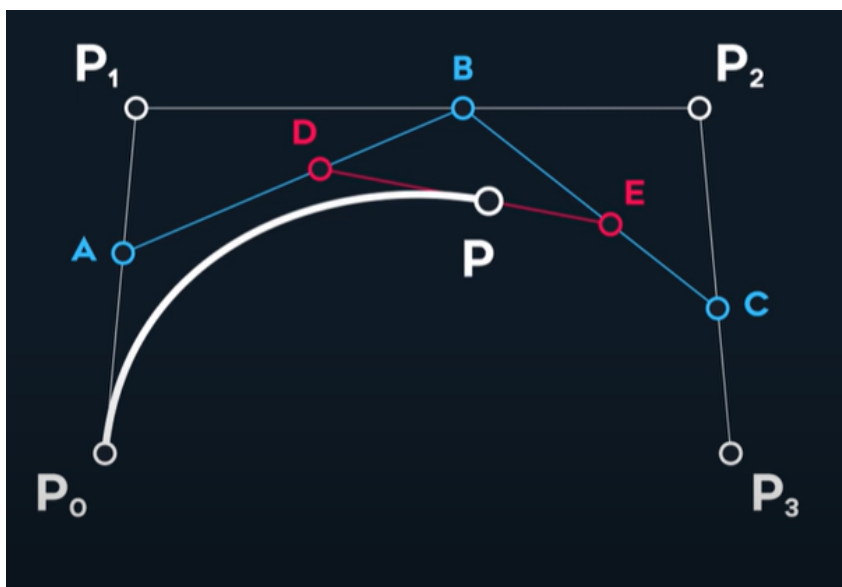
lerp(P1,P2,t) -> the resulting point will be B

lerp(P2,P3,t) -> the resulting point will be C

lerp(A,B,t) -> the resulting point will be D ----- Solving Quadratic Bezier here!

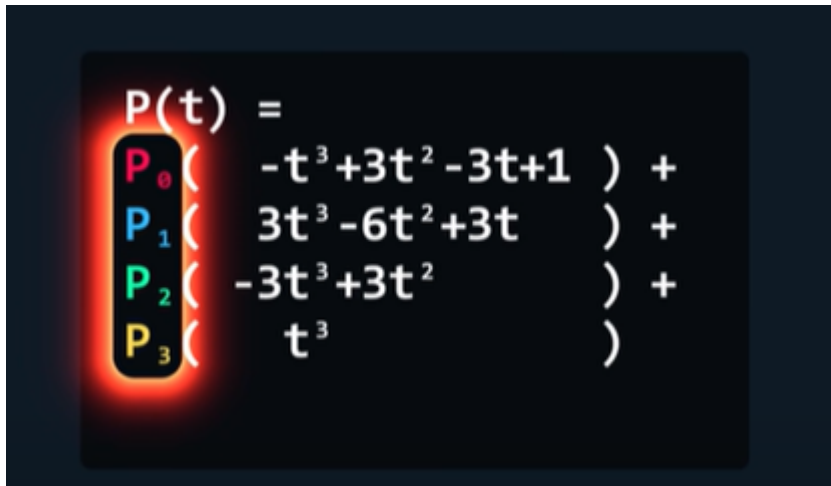
lerp(B,C,t) -> the resulting point will be E

lerp(D,E,t) ---> Hurray!!! Point P describing the curve!!! Check figure below.



$$\mathbf{B}(t) = (1 - t)^3 \mathbf{P}_0 + 3(1 - t)^2 t \mathbf{P}_1 + 3(1 - t) t^2 \mathbf{P}_2 + t^3 \mathbf{P}_3,$$

General Equation of cubic! We can visualize it based on the Ps



$$\begin{aligned} \mathbf{P}(t) = & \mathbf{P}_0(-t^3 + 3t^2 - 3t + 1) + \\ & \mathbf{P}_1(3t^3 - 6t^2 + 3t) + \\ & \mathbf{P}_2(-3t^3 + 3t^2) + \\ & \mathbf{P}_3(t^3) \end{aligned}$$

You can already see the matrix here :, if we factor out t as well we'll get *drum rolls*

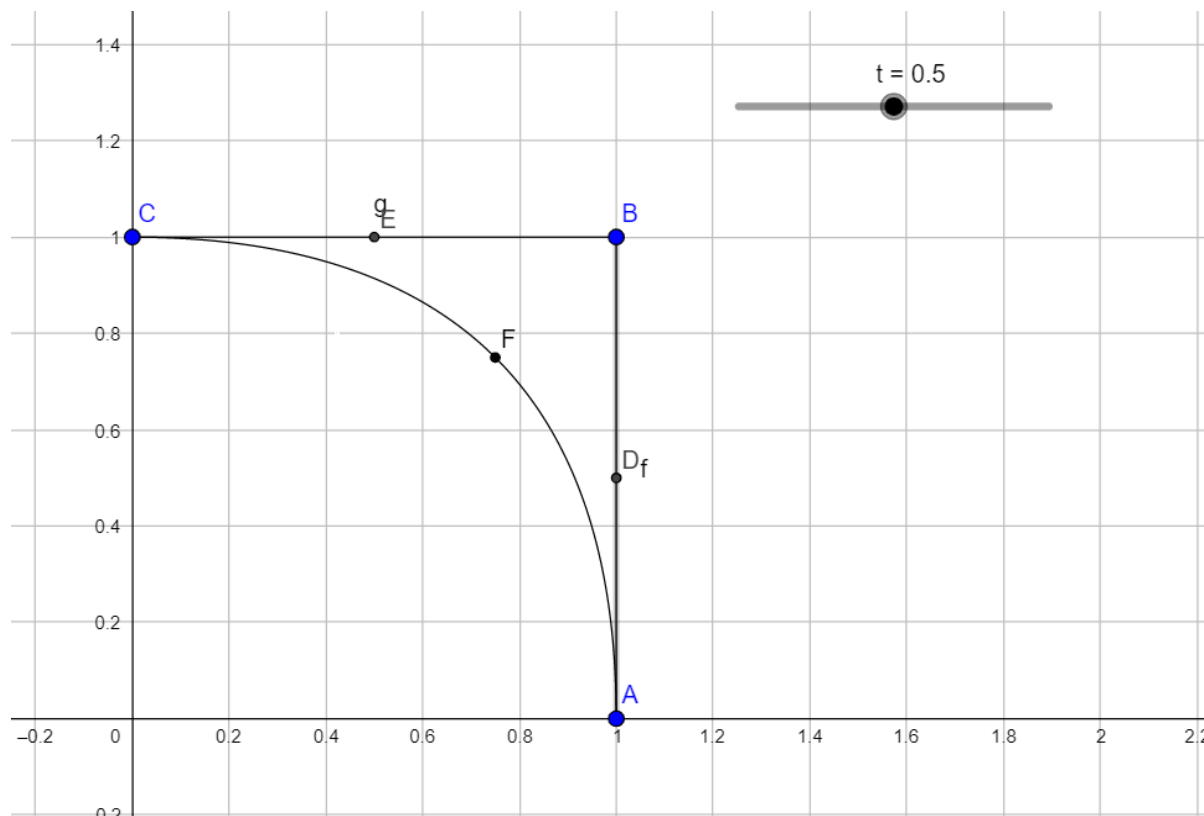
$$\mathbf{p}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

Is it easier to visualise now?

8. Consider a Bezier curve with control points $(1,0)$, $(1,1)$, $(0,1)$. Using the matrix you de-rived for question 8, what is the mid point on the curve?

As many of you did in the workshops, let's draw the curve, as you can see intuitively, the midpoint is clearly not $M(1/2, 1/2)$. So what do we do now? Look at question 7 and pluck $t = 0.5$ in the equation.

N.B. AGAIN, it is not t as in time or midway points of lines in between control points (they are more like weights, vectors if you like) it is the moment where all the LERPs recursively converged to the middle of the curve at $t = 0.5$!



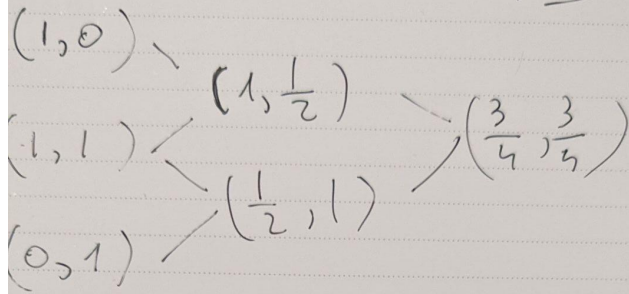
$$\begin{aligned}
 (1 - 0.5)^2 P_0 + 2(1 - 0.5) * 0.5 * P_1 + 0.5^2 * P_2 &= \\
 0.25 * P_0 + 0.5 * P_1 + 0.25 * P_2 &= \text{(replace with coordinates)} \\
 0.25 * [1, 0] + 0.5 * [1, 1] + 0.25 * [0, 1] &= \\
 [1/4, 0] + [1/2, 1/2] + [0, 1/4] &= [3/4, 3/4] \text{ -- Result for } M(3/4, 3/4)
 \end{aligned}$$

ALTERNATIVELY use De Casteljau's algorithm -- which is essentially decomposing the lerps at $t=0.5$ like this

$$t = 0.5$$

lerp \bar{I}

lerp \bar{u}



Above, sum in pairs and multiply
by $t = 0.5$