De Castlejau's Algorithm and the beauty of Animation mathematics

When people think of animation they usually think of Pixar, Cartoon Network, Disney Channel etc. Not many people would ever associate animation with mathematics but in this essay I will be arguing that the beauty of modern animation is stemmed from mathematical principles - in particular I will be discussing the beauty of mathematical animation curves used to model motion and de Castlejau's algorithm. The incorporation of mathematics into animation has led to its golden age. Animation entities come to life through meticulous gradient and shading details, intricate body movements and this is all the beauty of animation. So, where does the mathematics all come in?

Introduction:
In an early interview, co-founder and president of Pixar Animation Studios stated that the beauty of Pixar's early animation works was based on his work on displaying bicubic patches using a recursive subdivision algorithm (Kennedy and Haunsperger, 2001). Computer animation has many subtopics which makes up its foundation including advanced mathematical topics such as Fourier transformations and Complex Analysis but an elementary and practical understanding of this subject can be demonstrated through bézier curves and de Casteljau's algorithm.

What is de Casteljau's algorithm?
The origin of de Casteljau's algorithm and Bézier curves was actually rooted from research at car manufacturers Citroen and Renault. These days its application spans to computer aided design and typefacing (Long, C. and Norton, V., 1980). Bézier curves are generally defined parametrically under the following equation:

$$C(t) = \sum_{i=0}^{n} b_{i,j}(t) P_i$$

Where $b_{i,j}$ is defined as a Bernstein Polynomial which uses a binomial coefficient.

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

Where

$$0 \leq t \leq 1$$

Key:

P_i are the control points of a Bézier curve, which enables polygons to be formed (by connecting the control points P_0 to P_n.

The parametric nature of Bézier curves allows for the recursive linear interpolation by manipulating the t parameter (under the constraint that t is more or equal to zero and less than or equal to 1).

Bézier curves can then be recursively subdivided using de Casteljau's algorithm in order to create refined polynomials. It is a numerically stable way to calculate Bernstein polynomials, which means that with every recursive execution of this algorithm the significance of any input errors lessen so that they are negligible in the final output (Macura, W.K.).

They also have additional interesting mathematical properties such as the fact that the starting and ending sections of a control polygon form tangents with the start and end of a Bézier curve. This will be demonstrated further in a simulation/visualisation of de Casteljau's algorithm at work written using the programming language Python.

Bézier curves also have a convex hull property, which means that its curves are contained within a control polygon. This property is useful in aiding the ease of modifying and adjusting curves as desired using graphical transformations.

Modelling using de Casteljau's algorithm and programming techniques

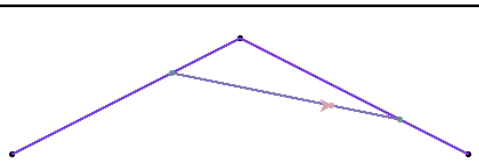
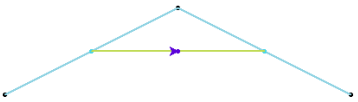
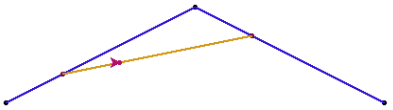
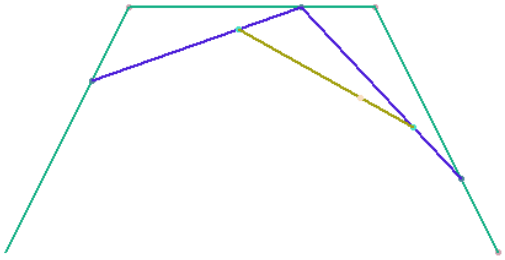
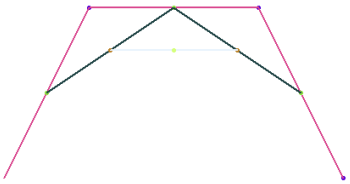
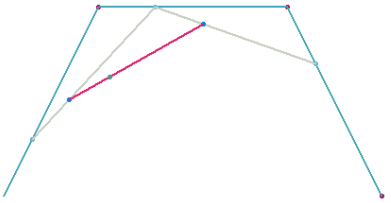
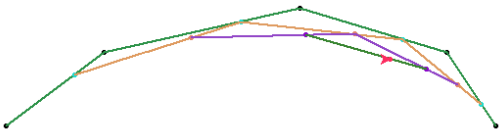
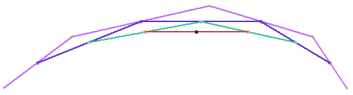
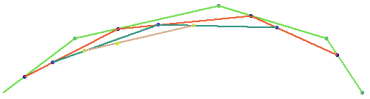
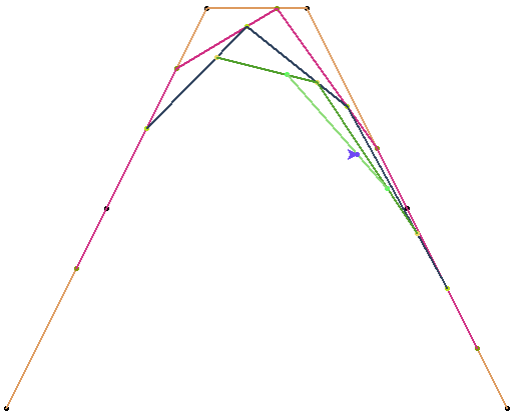
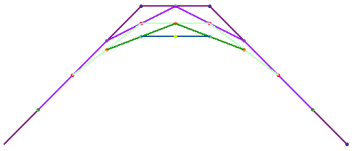
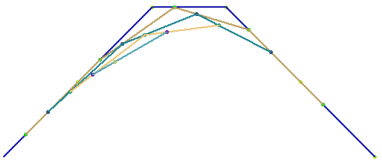
Python Code for deCasteljau's algorithm:

```
def deCasteljau(x1, y1, x2, y2):
    #calculate t point between these coordinates
    t_x = (x1 * (1-tFunction)) + (x2*tFunction)
    t_y = (y1 * (1-tFunction)) + (y2*tFunction)
    t_plot = (t_x, t_y)
    t_coords.append(t_plot)
    return t_plot
```

The full code I have written to produce the graphical results can be found at:
https://github.com/jade-bejide/deCasteljau

Results

Number of Control Points/t-value

	0.7	0.5	0.3
3			
4			
5			
6			

Observations

As demonstrated by the above table, increasing the number of control points increases the detail of the eventual bézier curve and t-values closer to zero means that the tip of the curve will be further away from the last control point and vice versa as the t-value is increased, up to one. These facts are beneficial in demonstrating the ease of transformation from one bézier curve mockup to another.

Changing the position of the control points and applying transformations to these Bézier curves can also manipulate the shape of the curve.

Computational Limitations of de Casteljau's algorithm

As aforementioned, de Casteljau's algorithm is a recursive subdivision algorithm, as a result, it has computational limitations.

One key example of the limitations of de Casteljau's algorithm is that there can only be a limited number of control points as a program executing de Casteljau will eventually reach a stack overflow. This is a phenomenon that occurs in Computer Science of which when a computer executes a recursive algorithm, it holds call stacks - a static data structure (in terms of size) that keeps a record of the information of subroutines and other subroutines it may have called during the recursive execution of an algorithm. This call stack size is typically 1MB in size on Windows workstations and can be up to 8MB on some Unix powered computers (10.10 — The stack and the heap | Learn C++, 2021).

This fact naturally creates a synoptic link between this topic of Bézier curves and de Casteljau's algorithm and algorithmic complexity and P vs NP, topics of research in Discrete Mathematics and Computer Science.

Bézier Curves in Affine Space

Affine space is a geometric set of values of which mappings between sets preserves the structure of the original set, adding to the properties of Euclidean space, which is the geometry most people are familiar with. The fact the structure of a set is preserved is a necessary part of the transformation of Bézier curves, modelling Bézier curves in affine space means that the ratio of lengths for line segments are preserved.

The structure of Bézier curves means that it can be transformed to manipulate the curve whilst maintaining collinearity and the ratio of distances between each control point (Weisstein, E.W.). To put this in simpler terms, Bézier curves are invariant in affine space.

Applications

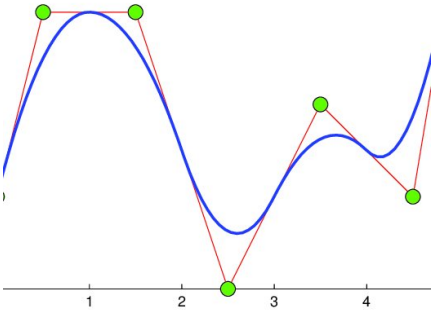
This essay discusses the use of Bézier curves in animation particularly, to create character and object motion.

Motion

As aforementioned, the purpose of Bézier curves is to quickly (using computational methods) build smooth curves by plotting control points. This enables animation engineers to create smooth motion paths for objects to improve overall cinematic experience and closeness to real world motion - as opposed to linear motion. The nature of Bézier curves

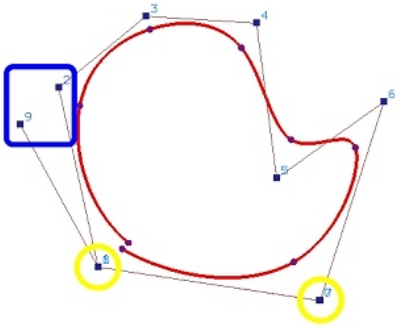
can be manipulated to demand in order to create different motion path shapes. Examples of such path shapes are defined in these key terms:

Open Spline This is the default shape of Bézier curves which can be edited by adjusting the control points.

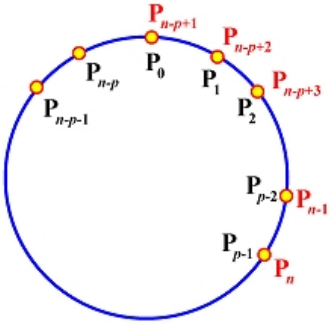


A quadratic (p = 2) B-spline curve with a uniform open knot vector Ξ = {0, 0, 0, 1, 2, 3, 4, 5, 5, 5}.

Closed Spline Where the start and ending control points are the same, forming a closed path.



Circle A subset of closed splines of which a circle/ellipse is created



Conclusion

So, to conclude, this was a brief and basic discussion introducing Bézier Curves and de Casteljau's algorithm. This topic covers many fields of research from computer graphics and Computer Science and a few different geometric spaces. Feel free to download and explore my coded application of the algorithm, to test its recursive limits and draw potential path shapes.

References:

B-spline Curves: Closed Curves. Available at: <https://pages.mtu.edu/~shene/COURSES/cs3621/NOTES/spline/B-spline/bspline-curve-closed.html> [accessed 26/03/21]

DEVELOPMENT OF ISOGEOMETRIC FINITE ELEMENT METHODS - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/A-quadratic-p-2-B-spline-curve-with-a-uniform-open-knot-vector-X-0-0-0-1-2_fig1_277405448 [accessed 26 Mar, 2021]

Farin, Gerald & Hansford, Dianne (2000). *The Essentials of CAGD*. Natic, MA: A K Peters, Ltd. ISBN 1-56881-123-3

Kennedy, S. and Haunsperger, D., 2001. Math Makes the Movies An Interview with Edwin Catmull. *Math Horizons*, *9*(2), pp.5-7.

Learncpp.com. 2021. *10.10 — The stack and the heap | Learn C++*. [online] Available at: <https://www.learncpp.com/cpp-tutorial/the-stack-and-the-heap/> [Accessed 26 March 2021].

Long, C. and Norton, V., 1980. Bezier Polynomials in Computer-Aided Geometric Design. *The Two-Year College Mathematics Journal*, *11*(5), pp.320-325.

Macura, Wiktor K. "Numerical Stability." From MathWorld--A Wolfram Web Resource, created by Eric W. Weisstein. https://mathworld.wolfram.com/NumericalStability.html

Weisstein, Eric W. "Affine Transformation." From MathWorld--A Wolfram Web Resource. https://mathworld.wolfram.com/AffineTransformation.html

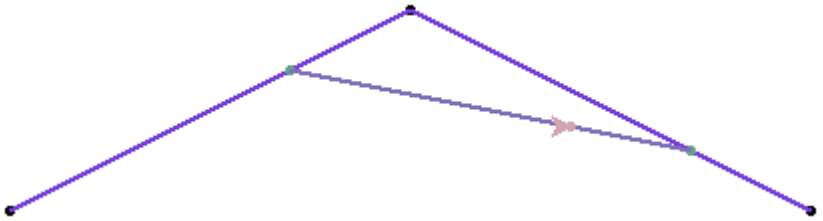
Links:

https://books.google.co.uk/books?hl=en&lr=&id=ogz5FjmiqlQC&oi=fnd&pg=PA1&dq=visual+complex+analysis&ots=lYPuo5WbgK&sig=1UWZUNIJmmkqVQifeHy8WvDfU44&redir_esc=y#v=onepage&q=visual%20complex%20analysis&f=false

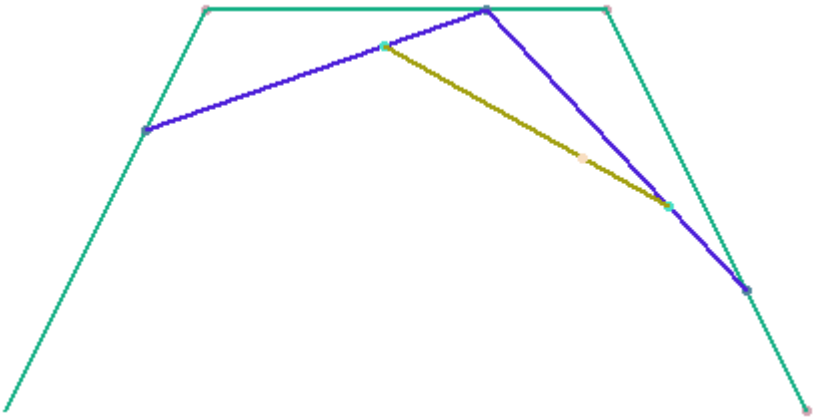
Appendix

1. Using a t value of 0.7

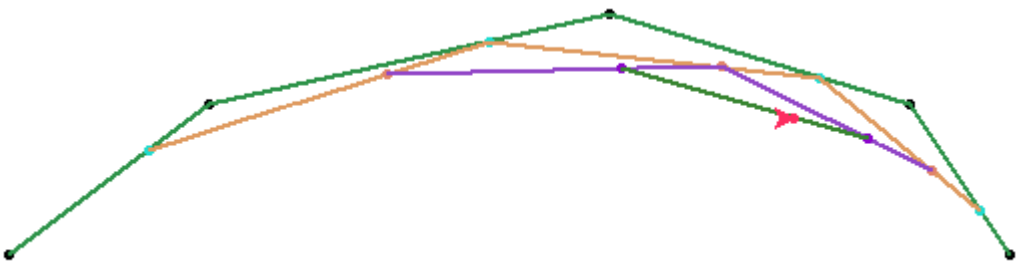
3-control points



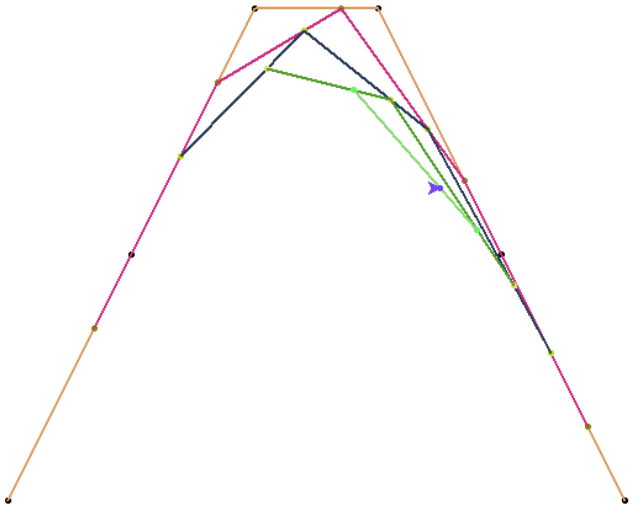
4-control points



5-control points

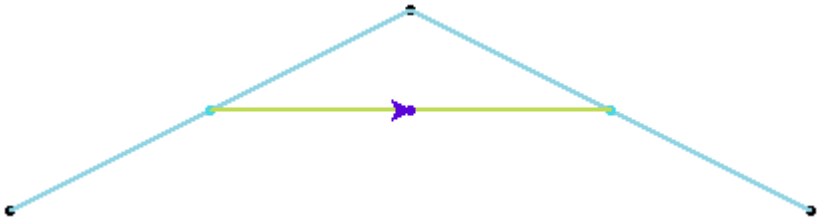


6-control points

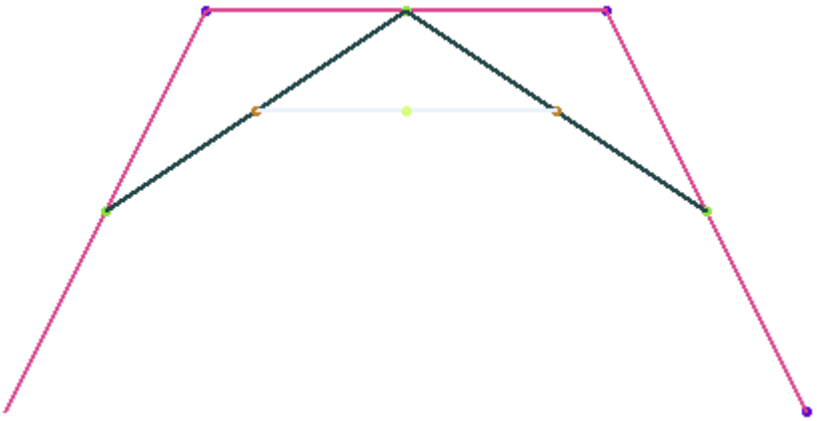


2. Using a t value of 0.5

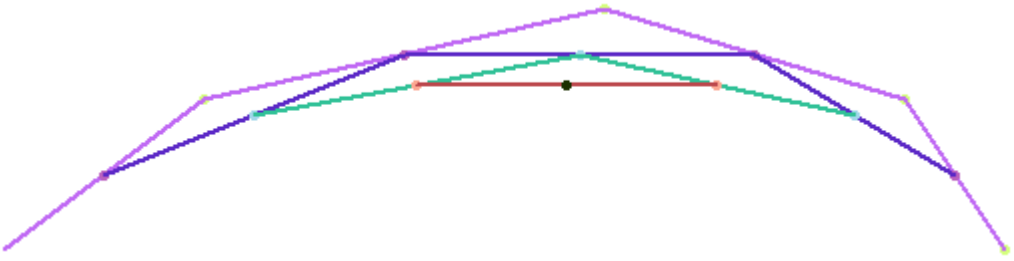
3-control points



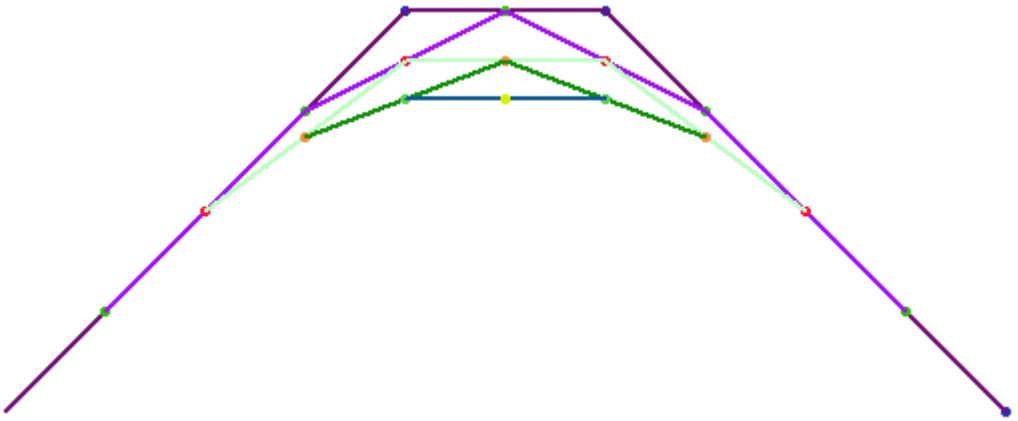
4-control points



5-control points

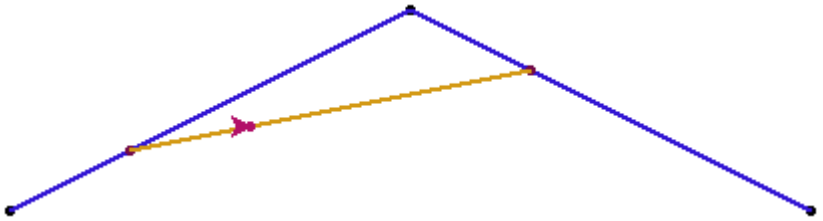


6-control points

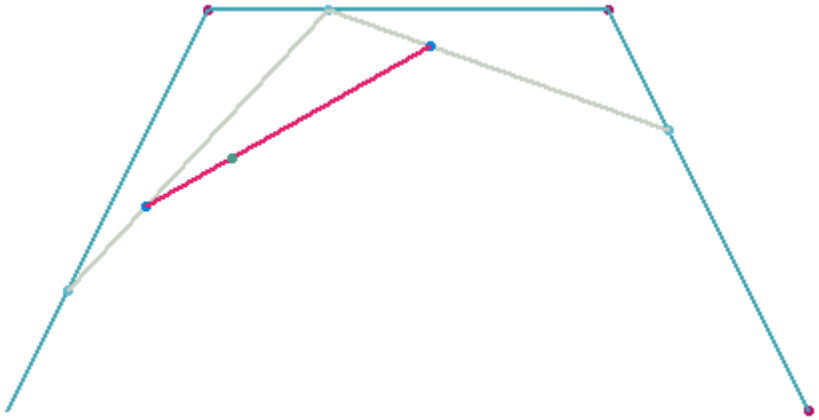


3. Using a t value of 0.3

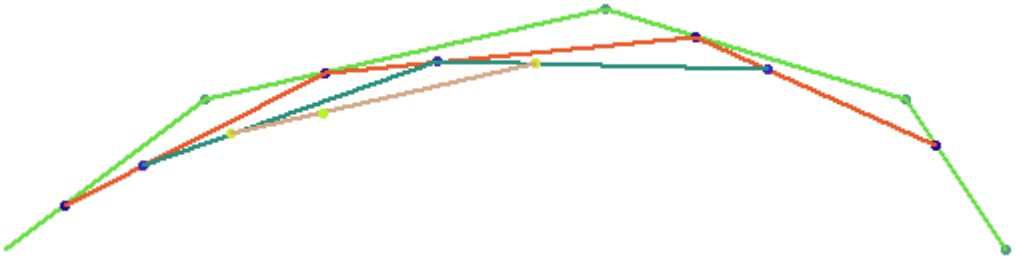
3-control points



4-control points



5-control points



6-control points