Computer Graphics

P. Healy

CS1-08
Computer Science Bldg.
tel: 202727
patrick.healy@ul.ie

Spring 2021–2022

Outline

Outline

Interpolation

- Not all "curves" are as regular as lines, circles or ellipses
- What happens when we want to do the best job we can of fitting a curve to a set of points?
- Fitting a curve **exactly** to a set of points, $S = \{(x_i, y_i)\}$ is called **interpolation**
- Idea: find a polynomial function $y = f(x)$ so that $y_i = f(x_i)$ for all $(x_i, y_i) \in S$
- Interpolation
- If $|S| = m$ then it is a fact that there is a polynomial of degree $m - 1$ that fits exactly
- Aside: one of the most popular methods for fitting curves is the **least-squares polynomials** method
  - this is called **curve fitting**
  - this is **not**

Curve Fitting

- In graphics the method most frequently used for fitting a curve to a set of points is the **spline**
- Curves are “created” piecewise by splicing together low-order (“not very complicated”) polynomials that fit subpaths of the points
- An example of interpolation

Bézier Curves; §14-8

- The **Bézier curve** is the most commonly used method for curve **fitting** (may not go through all points) Wikipedia page
- They are named after one of their co-inventors who worked in Renault's car design division in 1960's
- Bézier curves are **parametric** curves: a $\boxed{}$ of degree n in the $\boxed{}$ t
- Parameter t, $0 \leq t \leq 1$ is a measure of how "far along the path" we are in fitting the curve to points $p_0, p_1, \ldots, p_n$ animation

Bézier Curves

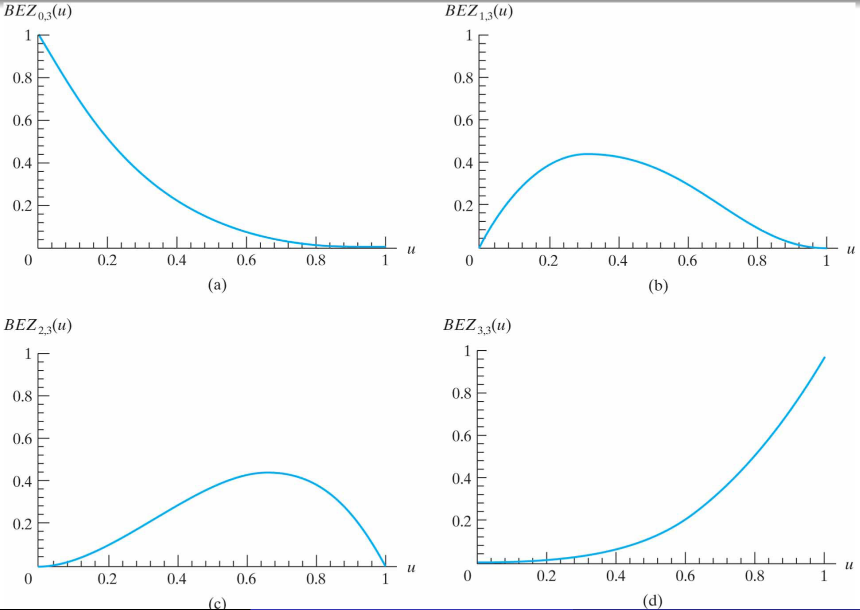
- Given $n+1$ points, $p_0, \ldots, p_n$, at "time t" Bézier curves rely crucially on breaking the curve into two smaller curves
- If we call $B_{0,\ldots,n}(t)$ the $n+1$-point curve, then we can show that $B(t)$ is recursively defined as an interpolation of two n-point curves

$$B_{0,\ldots,n}(t) = (1-t)B_{0,\ldots,n-1}(t) + tB_{1,\ldots,n}(t)$$

- $$B_{0,\ldots,n}(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i \, p_i = \sum_{i=0}^{n} B_i(t) \, p_i$$

  We call $B_i(t) = \binom{n}{i}(1-t)^{n-i}t^i$ the ith **blending** function; this is the $\boxed{}$ attached to p_i as t varies
- Fig over shows (L-R) $B_0(t), B_1(t), B_2(t), B_3(t)$ when $n=3$

Bézier Curves

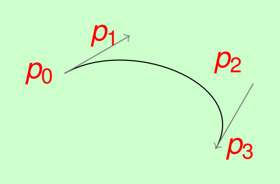


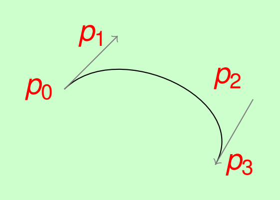
Bézier Curves

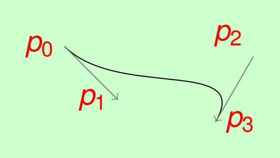
- Some consequences of definition of $B(t)$
  - $B(t)|_{t=0} = p_0$ and $B(t)|_{t=1} = p_n$ (curve always goes through end points)
  - The start (end) of the curve is tangent to the line $\overline{p_0 p_1}$ ($\overline{p_{n-1} p_n}$)
  - In general the curve does **not** interpolate (go through) any of the “between points”; this is why they are called **control** points
  - If we call the **convex hull** of the set of points the area enclosed by the perimeter of the points, then the curve never strays outside this bounding area
- We will only consider cubic BCs since higher order curves are more time-intensive to compute

Some Bézier curves

For cubic polynomials ($n - 1 = 3$) we work with sets of $n = 4$ points



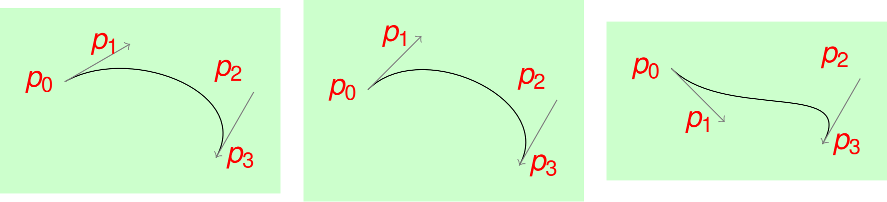




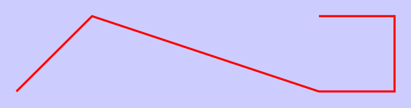
The curve that is created from the path of points will have as tangent at its start (end) point the line that joins the first (last) two points of the path.

Some Bézier curves

For cubic polynomials ($n - 1 = 3$) we work with sets of $n = 4$ points

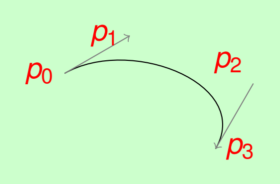


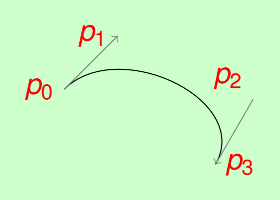
The curve that is created from the path of points will have as tangent at its start (end) point the line that joins the first (last) two points of the path.

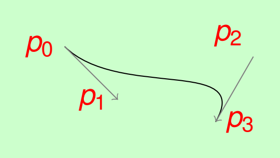


Some Bézier curves

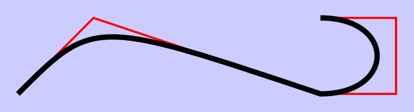
For cubic polynomials ($n - 1 = 3$) we work with sets of $n = 4$ points







The curve that is created from the path of points will have as tangent at its start (end) point the line that joins the first (last) two points of the path.



Computing a Linear Bézier Curve

- The parameter $0 \leq t \leq 1$ can be thought of as "how far along the path from start to finish we are"
- With p_0 as start and p_1 as end our position at "time" t is

$$\begin{aligned} p(t) &= p_0 + t(p_1 - p_0) \\ &= (1 - t)p_0 + tp_1 \end{aligned}$$

- (This is a very frequently used formula and is often referred to as the **convex combination** of p_0 and p_1)
- Wikipedia demo here

Computing a Cubic Bézier Curve

- Khan Academy demo here
- At any "time" t from points $p_0, \ldots, p_3$ we compute the distance travelled along each of the three lines

$$
\begin{aligned}
m_1 &= (1-t)p_0 + tp_1 \\
m_2 &= (1-t)p_1 + tp_2 \\
m_3 &= (1-t)p_2 + tp_3
\end{aligned}
$$

- We do the same again with the two lines that join the points m_1, m_2, m_3 to get m_{12} and m_{23}

$$
\begin{aligned}
m_{12} &= (1-t)m_1 + tm_2 \\
&= (1-t)((1-t)p_0 + tp_1) + t((1-t)p_1 + tp_2) \\
&= (1-t)^2 p_0 + 2(1-t)tp_1 + t^2 p_2 \\
m_{23} &= (1-t)^2 p_1 + 2(1-t)tp_2 + t^2 p_3
\end{aligned}
$$

Computing a Cubic Bézier Curve

- Khan Academy demo here
- At any "time" t from points $p_0, \ldots, p_3$ we compute the distance travelled along each of the three lines

$$\begin{aligned} m_1 &= (1-t)p_0 + tp_1 \\ m_2 &= (1-t)p_1 + tp_2 \\ m_3 &= (1-t)p_2 + tp_3 \end{aligned}$$

- We do the same again with the two lines that join the points m_1, m_2, m_3 to get m_{12} and m_{23}

$$\begin{aligned} m_{12} &= (1-t)m_1 + tm_2 \\ &= (1-t)((1-t)p_0 + tp_1) + t((1-t)p_1 + tp_2) \\ &= (1-t)^2 p_0 + 2(1-t)tp_1 + t^2 p_2 \\ m_{23} &= (1-t)^2 p_1 + 2(1-t)tp_2 + t^2 p_3 \end{aligned}$$

Computing a Cubic Bézier Curve

- Khan Academy demo here
- At any "time" t from points $p_0, \ldots, p_3$ we compute the distance travelled along each of the three lines

$$\begin{aligned} m_1 &= (1-t)p_0 + tp_1 \\ m_2 &= (1-t)p_1 + tp_2 \\ m_3 &= (1-t)p_2 + tp_3 \end{aligned}$$

- We do the same again with the two lines that join the points m_1, m_2, m_3 to get m_{12} and m_{23}

$$\begin{aligned} m_{12} &= (1-t)m_1 + tm_2 \\ &= (1-t)((1-t)p_0 + tp_1) + t((1-t)p_1 + tp_2) \\ &= (1-t)^2 p_0 + 2(1-t)tp_1 + t^2 p_2 \\ m_{23} &= (1-t)^2 p_1 + 2(1-t)tp_2 + t^2 p_3 \end{aligned}$$

Computing a Cubic Bézier Curve

- Doing this one last time gives us

$$
\begin{aligned}
B(t) =& (1-t)m_{12} + tm_{23} \\
=& (1-t)^3 p_0 + 2(1-t)^2 t p_1 + (1-t)t^2 p_2 + \\
& (1-t)^2 t p_1 + 2(1-t)t^2 p_2 + t^3 p_3 \\
=& (1-t)^3 p_0 + 3(1-t)^2 t p_1 + 3(1-t)t^2 p_2 + t^3 p_3 \\
=& \sum_{i=0}^{3} \binom{3}{i} (1-t)^{3-i} t^i \, p_i = \sum_{i=0}^{3} B_i(t) \, p_i
\end{aligned}
$$

- Also, if we call above the , $B_{0,\ldots,n}(t)$, then we can show that $B(t)$ is recursively defined as an interpolation of two n-point curves

$$B(t) = B_{0,\ldots,n}(t) = (1-t)B_{0,\ldots,n-1}(t) + tB_{1,\ldots,n}(t)$$

(see Wikipedia animations – they use this method)

Computing a Cubic Bézier Curve

- Doing this one last time gives us

$$
\begin{aligned}
B(t) =& (1-t)m_{12} + tm_{23} \\
=& (1-t)^3 p_0 + 2(1-t)^2 t p_1 + (1-t)t^2 p_2 + \\
& (1-t)^2 t p_1 + 2(1-t)t^2 p_2 + t^3 p_3 \\
=& (1-t)^3 p_0 + 3(1-t)^2 t p_1 + 3(1-t)t^2 p_2 + t^3 p_3 \\
=& \sum_{i=0}^{3} \binom{3}{i} (1-t)^{3-i} t^i \, p_i = \sum_{i=0}^{3} B_i(t) \, p_i
\end{aligned}
$$

- Also, if we call above the , $B_{0,\ldots,n}(t)$, then we can show that $B(t)$ is recursively defined as an interpolation of two n-point curves

$$B(t) = B_{0,\ldots,n}(t) = (1-t)B_{0,\ldots,n-1}(t) + tB_{1,\ldots,n}(t)$$

(see Wikipedia animations – they use this method)