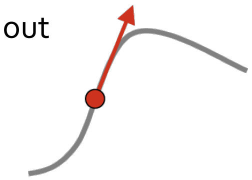
EECS 487: Interactive Computer Graphics

Lecture 36: Parametric surfaces

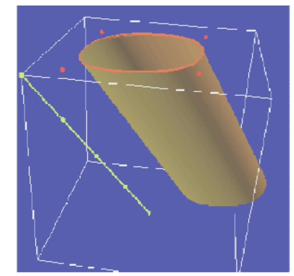
- Swept surfaces
- Geometric continuity
- Bézier curves and patches

Extruded/Swept Surfaces



Consider a curve in space as being swept out by a moving point: $\mathbf{p}(u) = [x(u)\ y(u)\ z(u)]^T$

- as we vary u the point moves through space
- the curve is the path taken by the point

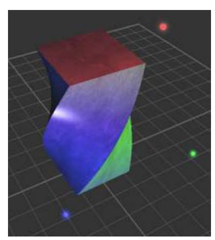


Similarly we can think of a surface: $\mathbf{s}(u, t) = [x(u, t)\ y(u, t)\ z(u, t)]^T$

- as being swept out by a profile curve along a trajectory curve
- the set of points visited by the curve during its motion defines the surface

Yu,Terzopoulos

General Sweep Surfaces



Trajectory path may be any arbitrary curve

The profile curve may be transformed as it moves along the path

- scaled, rotated with respect to path orientation, …

Example: surface $\mathbf{s}(u, t)$ is formed by a profile curve in the xy-plane $\mathbf{p}(u) = [x(u)\ y(u)\ 0\ 1]^T$ extruded along the z-axis:

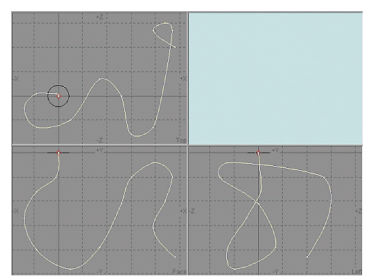
$\mathbf{s}(u, t)$: $\mathbf{T}(t)\mathbf{p}(u)$:

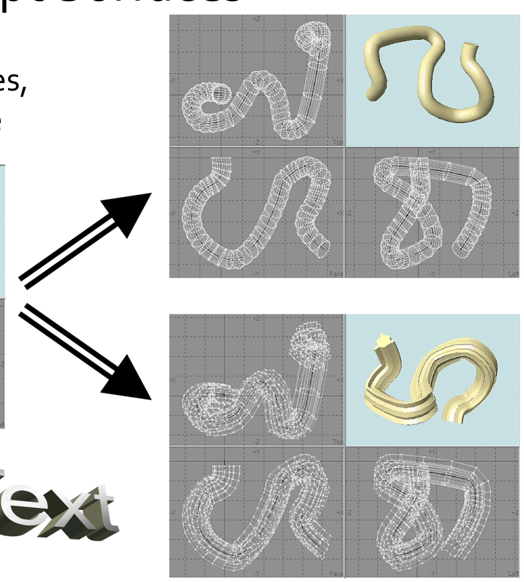
$$x(u, t) = x(u), y(u, t) = y(u), 0 \le u \le 1,$$
$$z(u, t) = t, z_{min} \le t \le z_{max}$$

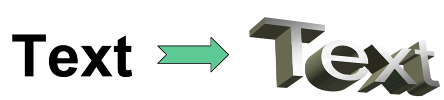
Fussell,Durand,Terzopoulos

Extruded/Swept Surfaces

Different profile curves, same trajectory curve





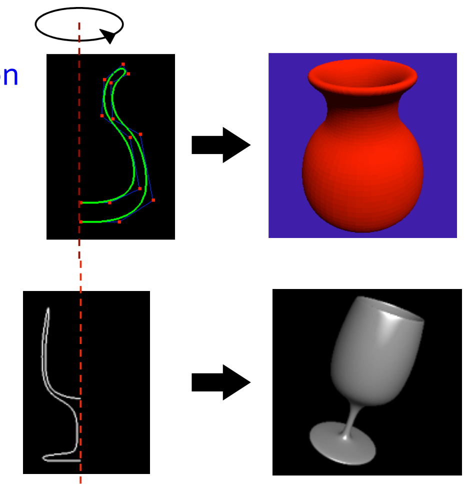


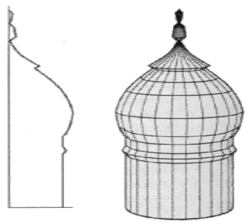
Schulze

Surfaces of Revolution

Use rotation around an axis instead of translation along a path

- or, extrusion where the trajectory curve is a circle
- $\mathbf{s}(u, t)$: $\mathbf{R}(t)\mathbf{p}(u)$



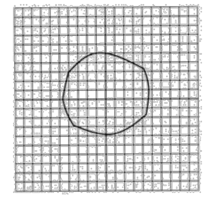


Schulze,Durand

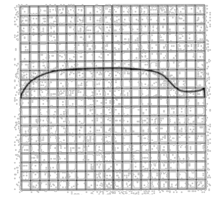
A Banana as a Generalized Cylinder

What we specify

- a mostly circular profile
- a spine for the banana
- a scaling function



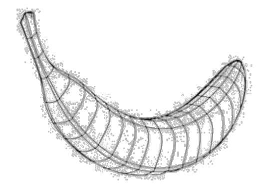
cross section



scaling function

Periodically along the spine

- place a cross section
- scale it appropriately
- connect to previous section

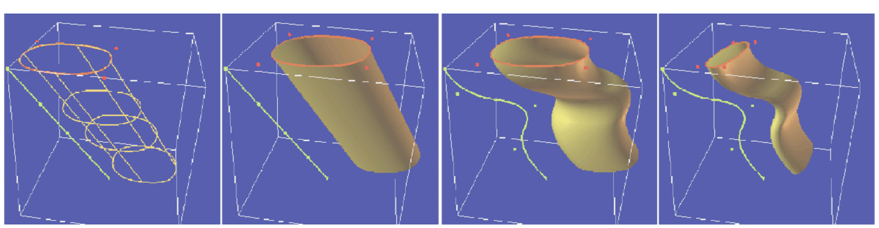


Yu,Snyder

General Sweep Surfaces

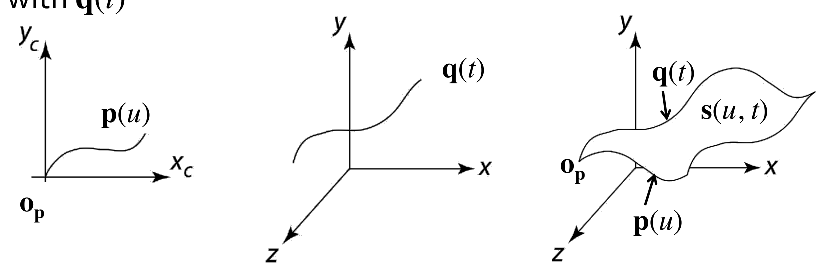
The trajectory curve is like a spine

- sweeping the profile curve "skins" a surface around the trajectory curve
- the shape of the spine controls the shape of the object
- nice for animation:
  - don't have to control the surface
  - just reshape the spine and the surface follows along



General Sweep Surfaces

For every point along $\mathbf{q}(t)$, lay $\mathbf{p}(u)$ so that $\mathbf{o_p}$ coincides with $\mathbf{q}(t)$



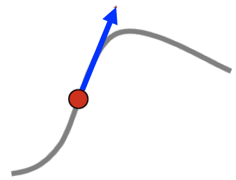
This gives us locations along $\mathbf{q}(t)$, how about orientation?

1. fixed or static: aligns $\mathbf{p}(u)$ with an axis
2. allows smoothly varying orientation that "follows" the orientation of $\mathbf{q}(t)$: how to specify the orientation of $\mathbf{q}(t)$?

Curless

Differential Geometry of Curves

Uses:
- define orientation of swept surfaces
- compute velocity of animation
- compute normals of surfaces
- analyze smoothness/continuity



Tangent:

The velocity of movement, 1st derivative with respect to t

$\mathbf{q}'(t) = (x'(t), y'(t), z'(t))$ or $\mathbf{q}'(t) \approx (\mathbf{q}(t+\Delta t) - \mathbf{q}(t))/\Delta t$

- $||\mathbf{q}'(t)||$ is the speed of movement
- normalized tangent $\mathbf{t}(t) = \mathbf{q}'(t)/||\mathbf{q}'(t)||$ is the direction of movement
- the numeric form of forward difference is useful if $\mathbf{q}(t)$ is a black box

The tangent provides us with the first of three orientations for swept surfaces

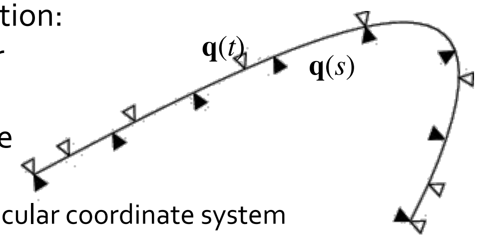
Durand

Arc Length Parameterization

For smooth motion, we want continuous 1st and 2nd derivatives with respect to time $d\mathbf{q}/dt$

But to describe shape, we could ask for continuity with respect to equal steps (arc length): $d\mathbf{q}/ds$

Arc length parameterization:
equal steps in parameter space s maps to equal distances along the curve
- intrinsic to shape of curve, not dependent on any particular coordinate system



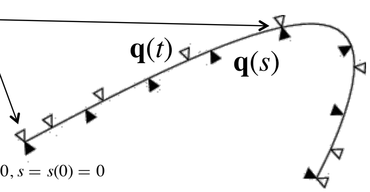
TP3, Hart

[Curless]

Arc Length Parameterization

If s is the length of curve from $\mathbf{q}(0)$ to $\mathbf{q}(t)$, $\mathbf{q}(s)$ can be expressed in terms of t:

$\mathbf{q}(s) = \{\mathbf{q}(t)\colon s(t) = s\}$, e.g.,



Unless moving at constant speed, arc length travelled is not proportional to passing time:
- i.e., equal steps in time (t) does not necessarily give equal distances in arc length (s)

$$s(t) = \int_0^t \left\|\mathbf{q}'(\tau)\right\| d\tau = \int_0^t \sqrt{x'(\tau)^2 + y'(\tau)^2 + z'(\tau)^2}\, d\tau \leftarrow$$

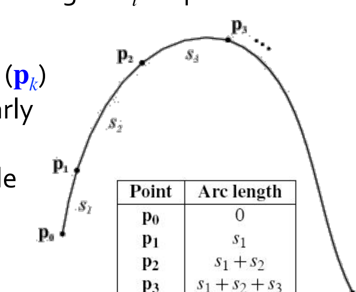
usually cannot be evaluated analytically

TP3, Hart, Curless

Arc Length by Linear Interpolation

Instead:
- pre-compute a set of variable arc lengths s_i for points on the curve using t parameterization
- to find the corresponding point ($\mathbf{p}_k$) on the curve for a given s_k, linearly interpolate the points of the 2 nearest arc lengths to either side s_i and s_{i+1}, $s_i \le s_k \le s_{i+1}$:

$$\mathbf{p}_k = \frac{s_{i+1} - s_k}{s_{i+1} - s_i}\,\mathbf{p}_i + \frac{s_k - s_i}{s_{i+1} - s_i}\,\mathbf{p}_{i+1}$$



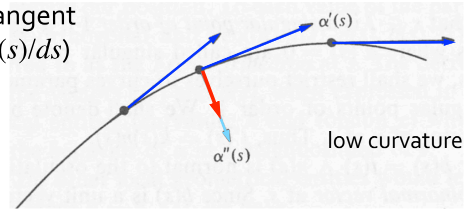
Point	Arc length
$\mathbf{p}_0$	0
$\mathbf{p}_1$	s_1
$\mathbf{p}_2$	$s_1 + s_2$
$\mathbf{p}_3$	$s_1 + s_2 + s_3$
...	...

TP3

Curvature and Normal

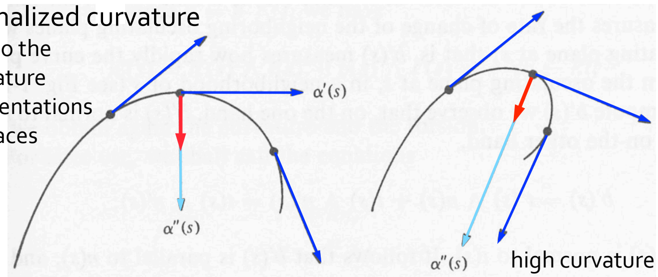
Curvature (κ): derivative of tangent with respect to arc length ($d\mathbf{t}(s)/ds$)

- how fast the curve pulls away from a straight line
- always orthogonal to tangent
- constant for a circle
- zero for a straight line



Normal: normalized curvature

- vector points to the center of curvature
- the 2[nd] of 3 orientations for swept surfaces

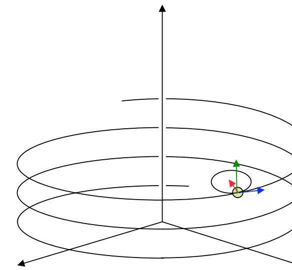


Durand

Torsion and Binormal

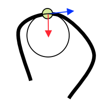
Torsion: deviation of the curve from the plane formed by the tangent and normal vectors

- zero for a plane curve
- binormal vector points to the winding direction of the space curve
- the 3[rd] of 3 orientations for swept surfaces



A curve is a 1D manifold in a space of higher dimension

- Plane (2D) curves, described by:
  - position, tangent, curvature
- Space/skew (3D) curves, described by:
  - position, tangent, curvature, torsion



Béchet

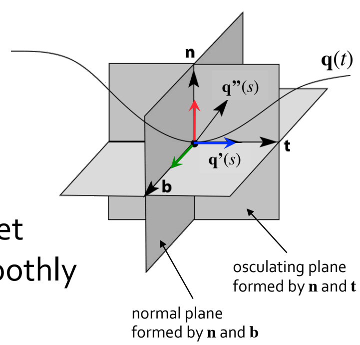
Frenet Frame

Given a curve $\mathbf{q}(t)$ we can attach a smoothly varying coordinate system consisting of three basis vectors (reparameterized to arc length):

- tangent: $\mathbf{t}(s) = \mathbf{q}'(s(t))$ (normalized)
- normal: $\mathbf{n}(s) = \mathbf{t}'(s)/||\mathbf{t}'(s)||$
- binormal: $\mathbf{b}(s) = \mathbf{n} \times \mathbf{t}$

Due to Jean Frédéric Frenet (1847) and Joseph Alfred Serret (1851)

As we move along $\mathbf{q}(t)$, the Frenet frame ($\mathbf{t}(s)$, $\mathbf{b}(s)$, $\mathbf{n}(s)$) varies smoothly

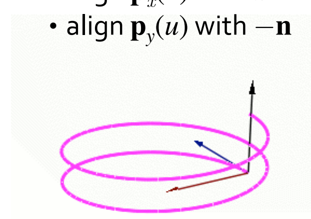


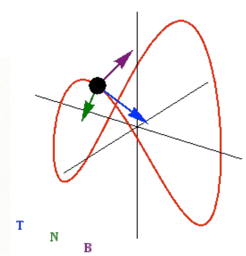
Curless

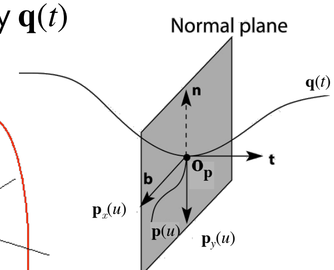
Frenet Swept Surfaces

Orient the profile curve $\mathbf{p}(u)$ using the Frenet frame of the trajectory $\mathbf{q}(t)$

- put $\mathbf{p}(u)$ in the normal plane of $\mathbf{q}(t)$
- place $\mathbf{o}_\mathbf{p}$ on $\mathbf{q}(t)$
- align $\mathbf{p}_x(u)$ with $\mathbf{b}$
- align $\mathbf{p}_y(u)$ with $-\mathbf{n}$



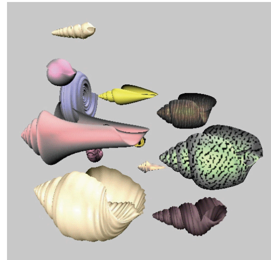




If $\mathbf{q}(t)$ is a circle, you get a surface of revolution exactly!

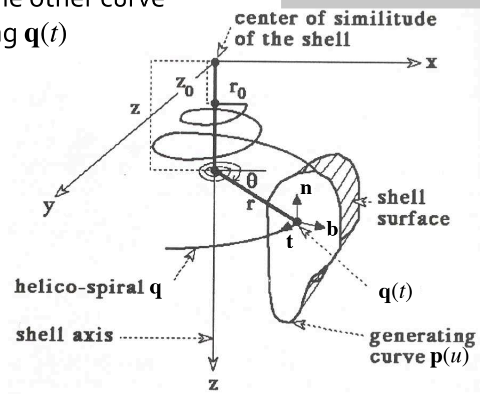
Curless,wikipedia

Variations



Several variations are possible:

- scale $\mathbf{p}(u)$ as it moves, possibly scaled to $||\mathbf{q}(t)||$
- morph $\mathbf{p}(u)$ into some other curve $\mathbf{f}(u)$ as it moves along $\mathbf{q}(t)$

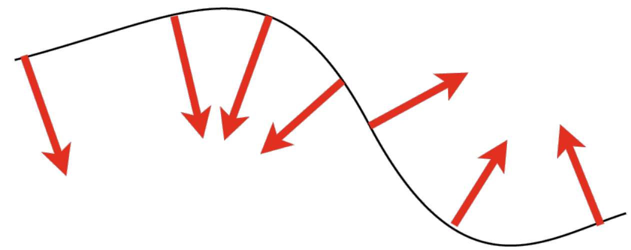


:unkhouser

Problems with Swept Surfaces

What happens at inflection points?

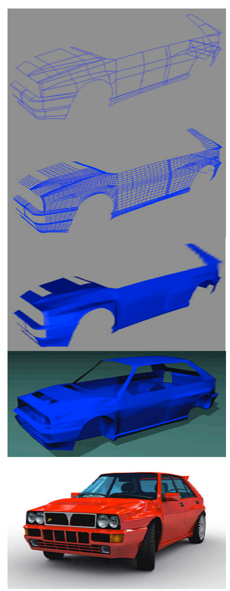
- curvature goes to zero
- then normal flips!
- resulting in a non-smooth swept surface



Also, difficult to avoid self-intersection

Curless,Fussell,Durand,Chenney

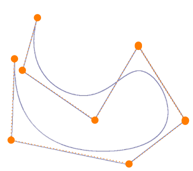
Free-form Surfaces



Swept surfaces are great, but we would like to represent "free-form" (asymmetric, irregular) curves and surfaces

We would also like to give model builders an intuitive control of a smooth shape

- specify objects with a few control points
- resulting in visually pleasing (smooth) objects



Schulze

Polynomial Surfaces

CAGD (Computer-Aided Geometric Design): area of CG dealing with free-form shapes

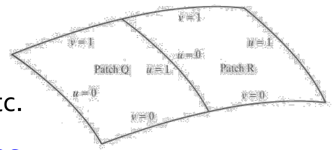
1960's:

- the need for mathematical representations of free-form shapes became apparent in the automotive and aeronautic industries
- Paul de Casteljau & Pierre Bézier independently developed the theory of polynomial curves & surfaces
- which became the basic tool for describing and rendering free-form shapes

Parametric Patches

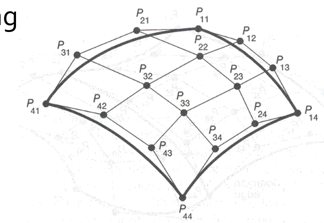
Parametric curves and surfaces give and require fewer degrees of control than polygonal meshes

- users control a few points
- program smoothly fills in the rest
- representation provides analytical expressions for normals, tangents, etc.



Surface is partitioned into patches:

- piecewise parametric surfaces (3D splines)
- each defined by control points forming a control net

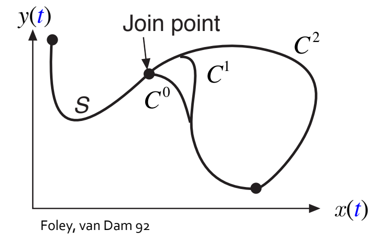


Most popular for modeling are Bézier, B-splines, and NURBS

- we'll study these as 2D splines first, then we'll use them as 3D patches

Funkhouser, FvDFH Fig. 11.44

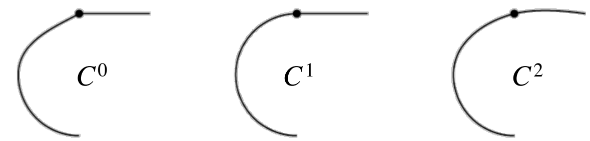
Measures of Joint Smoothness



Foley, van Dam 92

Parametric continuity:

- continuous by parameter t
- useful for trajectories
- 0^{th} order, C^0
  curve segments meet (join point): $\mathbf{f}_2(0) = \mathbf{f}_1(1)$
- 1^{st} order, C^1
  1^{st} derivatives, velocities, are equal at join point: $\mathbf{f}_2'(0) = \mathbf{f}_1'(1)$
- 2^{nd} order, C^2
  2^{nd} derivatives, accelerations, are equal at join point



Hodgins,Marschner

Joint Smoothness

C^0 continuous

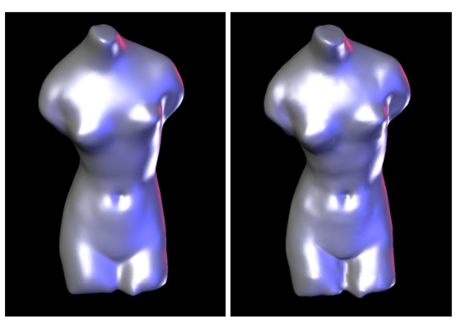
- curve/surface has no breaks/gaps/holes
- model is "watertight"

C^1 continuous

- model "looks smooth, no facets" (but sometimes looks like a lumpy potato)

C^2 continuous

- looks more polished: smooth specular highlights



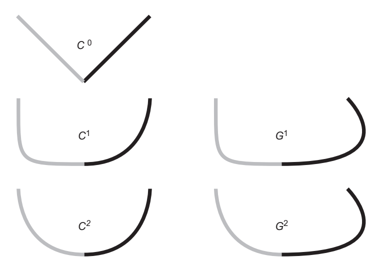
C^2 almost everywhere C^1 only

Durand

Measures of Joint Smoothness

Geometric continuity:

- continuous by parameter s (arclength)
- useful for defining shapes
- 1^{st} order, G^1
  1^{st} derivatives, tangents, are in the same direction and of proportional magnitude at join point: $\mathbf{f}_2'(0) = k\,\mathbf{f}_1'(1), k > 0$
- 2^{nd} order, G^2
  2^{nd} derivatives, curvatures, are proportional at join point



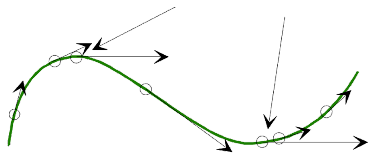
$\Rightarrow$ G^n continuity is usually a weaker constraint than C^n continuity (e.g., the "speed" along the curve does not matter)

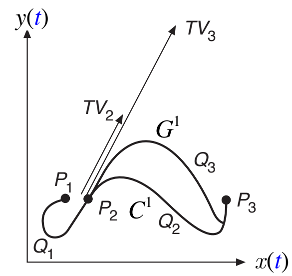
But neither form of continuity is guaranteed by the other

Shirley,Marschner

G^1 not C^1

G^1 but not C^1 when tangent direction doesn't change, but the magnitude changes abruptly

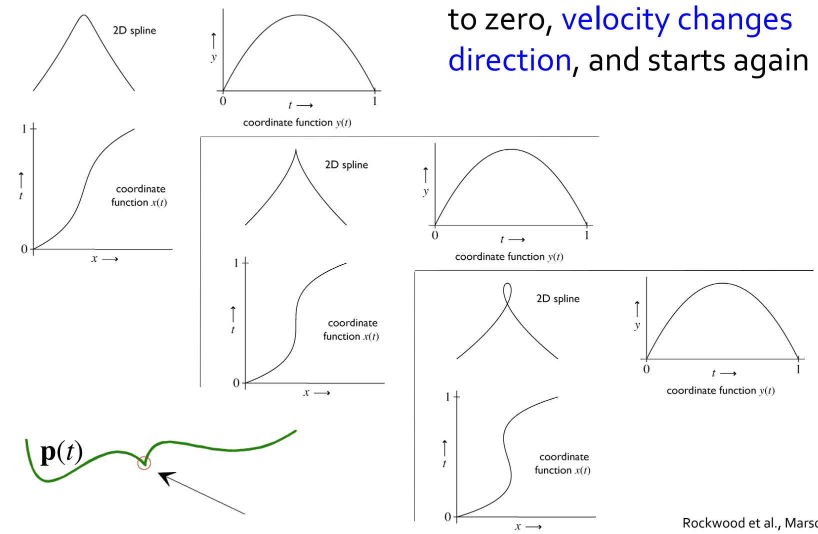




Rockwood et al., Marschner, FvD

C^1 not G^1

When the curve $\mathbf{p}(t)$ goes to zero, velocity changes direction, and starts again



Rockwood et al., Marschner

Cubic Splines

A representation of cubic spline consists of:

- four control points (why four?)
  - these are completely user specified
  - determine a set of blending functions

There is no single "best" representation of cubic spline:

Cubic	Interpolate?	Local?	Continuity	Affine?	Convex*?	VD*?
Hermite	✔	✔	C^1	✔	n/a	n/a
Cardinal (Catmull-Rom)	except endpoints	✔	C^1	✔	no	no
Bézier	endpoints	✗	C^1	✔	✔	✔
natural	✔	✗	C^2	✔	n/a	n/a
B-splines	✗	✔	C^2	✔	✔	✔

* n/a when some of the control "points" are tangents, not points

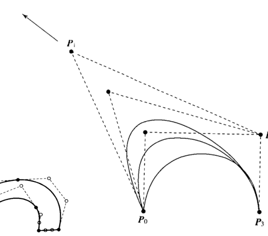
Bézier Curve

Named after Pierre Bézier, a car designer at Renault

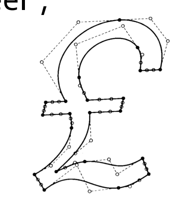
Independently developed by Paul de Casteljau at Citroën

Has an intuitive geometric "feel", easy to control

- common interface for creating curves in drawing programs
- used in font design (Postscript)



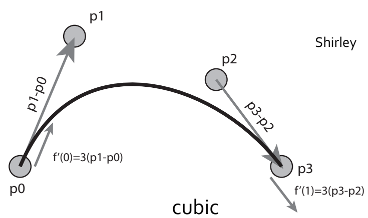
Watt



Foley & van Dam

Bézier Curve

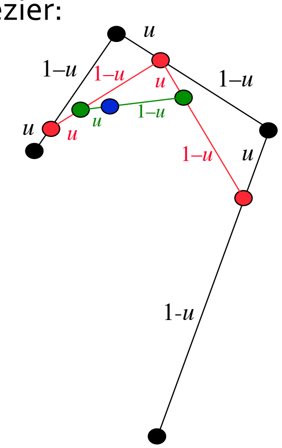
Uses an arbitrary number of control points (not just cubic)



- the first and last control points interpolate the curve
- the rest approximate the curve, control point i exerts the strongest attraction at $u = i/n,\ 1 \le i < n-1,\ 0 \le u \le 1$
- tangent at the start of the curve is proportional to the vector between the first and second control points
- tangent at the end of the curve is proportional to the vector between the second last and last control points
- the n-th derivative at the start (end) of the curve depends on the first (last) n+1 control points

de Casteljau Algorithm

A geometric evaluation scheme for Bézier: creates Bézier curve iteratively

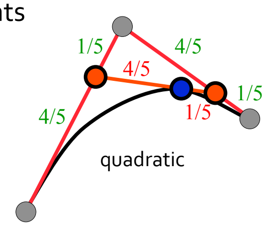


To compute $\mathbf{f}(u)$:

- connect adjacent control points with straight lines into a control polygon
- create the u interpolate points, $u \in [0,1]$, on these lines
  - at each iteration, there are n-1 such points
- connect the new points with straight lines
- repeat until only one new point is created

de Casteljau Quadratic Bézier

A quadratic Bézier curve has 3 control points



Let $u = 4/5$

$$\mathbf{p}_k = \mathbf{p}_0 + \tfrac{4}{5}(\mathbf{p}_1 - \mathbf{p}_0) = \tfrac{1}{5}\mathbf{p}_0 - \tfrac{4}{5}\mathbf{p}_1$$

$$\mathbf{q}_0 = \tfrac{1}{5}(\tfrac{1}{5}\mathbf{p}_0 + \tfrac{4}{5}\mathbf{p}_1) + \tfrac{4}{5}(\tfrac{1}{5}\mathbf{p}_1 + \tfrac{4}{5}\mathbf{p}_2)$$

Or more generally:

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

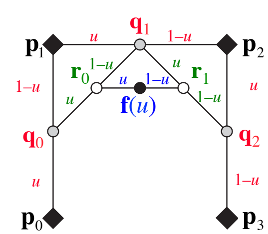
which is the quadratic Bézier curve:

$$\mathbf{f}(u) = (1-u)^2\mathbf{p}_0 + 2u(1-u)\mathbf{p}_1 + u^2\mathbf{p}_2$$

de Casteljau Cubic Bézier

Given four control points $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$, use de Casteljau algorithm to build a cubic Bézier curve $\mathbf{f}(u), 0 \le u \le 1$, with $\mathbf{p}_0 = \mathbf{f}(0)$, $\mathbf{p}_3 = \mathbf{f}(1)$ as shown:

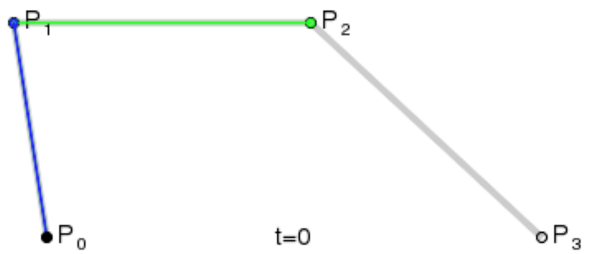
$$\mathbf{q}_0 = \mathbf{p}_0 + u(\mathbf{p}_1 - \mathbf{p}_0)$$
$$= (1-u)\mathbf{p}_0 + u\mathbf{p}_1$$
$$\mathbf{q}_1 = (1-u)\mathbf{p}_1 + u\mathbf{p}_2$$
$$\mathbf{q}_2 = (1-u)\mathbf{p}_2 + u\mathbf{p}_3$$
$$\mathbf{r}_0 = (1-u)\mathbf{q}_0 + u\mathbf{q}_1$$
$$\mathbf{r}_1 = (1-u)\mathbf{q}_1 +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



$$\boxed{\mathbf{f}(u) = (1-u)^3\mathbf{p}_0 + 3u(1-u)^2\mathbf{p}_1 + 3u^2(1-u)\mathbf{p}_2 + u^3\mathbf{p}_3}$$

de Casteljau Cubic Bézier

Draw out the curve by sweeping through time



$$\mathbf{f}(u) = (1-u)^3\mathbf{p}_0 + 3u(1-u)^2\mathbf{p}_1 + 3u^2(1-u)\mathbf{p}_2 + u^3\mathbf{p}_3$$

Then set:

$$\mathbf{f}'(0) = 3(\mathbf{p}_1 - \mathbf{p}_0)$$

$$\mathbf{f}'(1) = 3(\mathbf{p}_3 - \mathbf{p}_2)$$

[wikipedia]

Cubic Bézier Curve

Control points consist of endpoint interpolations and derivatives:

$$\begin{aligned}
\mathbf{f}(u) &= \mathbf{a}_0 + u^1\mathbf{a}_1 + u^2\mathbf{a}_2 + u^3\mathbf{a}_3\\
\mathbf{p}_0 = \mathbf{f}(0) &= \mathbf{a}_0 + 0^1\mathbf{a}_1 + 0^2\mathbf{a}_2 + 0^3\mathbf{a}_3\\
\mathbf{p}_3 = \mathbf{f}(1) &= \mathbf{a}_0 + 1^1\mathbf{a}_1 + 1^2\mathbf{a}_2 + 1^3\mathbf{a}_3\\
3(\mathbf{p}_1 - \mathbf{p}_0) = \mathbf{f}'(0) &= \mathbf{a}_1 + 2*0^1\mathbf{a}_2 + 3*0^2\mathbf{a}_3\\
\mathbf{p}_1 = \tfrac{1}{3}(\mathbf{f}'(0) + 3\mathbf{p}_0) &= \mathbf{a}_0 + \tfrac{1}{3}\mathbf{a}_1 + 0^2\mathbf{a}_2 + 0^3\mathbf{a}_3\\
3(\mathbf{p}_3 - \mathbf{p}_2) = \mathbf{f}'(1) &= \mathbf{a}_1 + 2*1^1\mathbf{a}_2 + 3*1^2\mathbf{a}_3\\
\mathbf{p}_2 = \tfrac{1}{3}(3\mathbf{p}_3 - \mathbf{f}'(1)) &= 1\mathbf{a}_0 + \tfrac{2}{3}\mathbf{a}_1 + \tfrac{1}{3}\mathbf{a}_2 - 0\mathbf{a}_3
\end{aligned}$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0\\ 1 & \tfrac{1}{3} & 0 & 0\\ 1 & \tfrac{2}{3} & \tfrac{1}{3} & 0\\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Constraint matrix

Basis matrix:

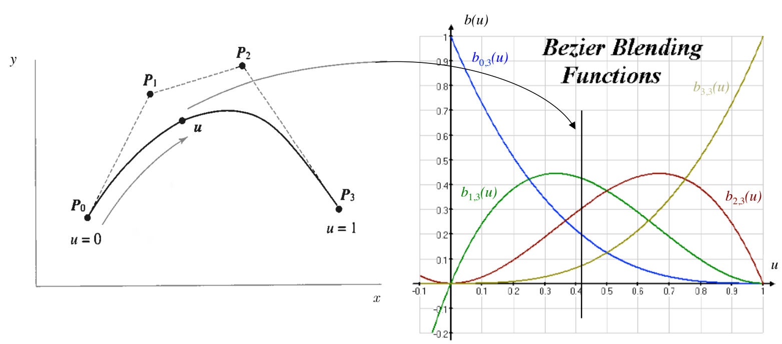
$$\mathbf{B} = \mathbf{C}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0\\ -3 & 3 & 0 & 0\\ 3 & -6 & 3 & 0\\ -1 & 3 & -3 & 1 \end{bmatrix}$$

Cubic Bézier Curve

$$\mathbf{uB} = \sum_{i=0}^{n} b_i(u)$$

Blending functions:

$$\begin{aligned}
\mathbf{f}(u) &= (1-u)^3\mathbf{p}_0 + 3u(1-u)^2\mathbf{p}_1 + 3u^2(1-u)\mathbf{p}_2 + u^3\mathbf{p}_3\\
&= \underbrace{(1-3u+3u^2-u^3)}_{b_{0,3}(u)}\mathbf{p}_0 + \underbrace{(3u-6u^2+3u^3)}_{b_{1,3}(u)}\mathbf{p}_1 + \underbrace{(3u^2-3u^3)}_{b_{2,3}(u)}\mathbf{p}_2 + \underbrace{u^3}_{b_{3,3}(u)}\mathbf{p}_3
\end{aligned}$$

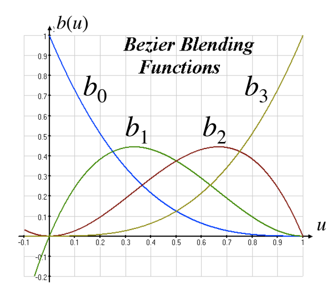


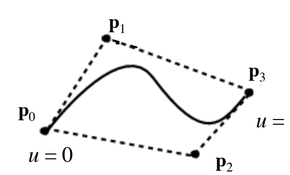
Watt, Hodgins

Cubic Bézier Properties

Properties:

- each b_i specifies the influence of $\mathbf{p}_i$
- convex hull: $\sum b_i = 1,\, b_i \geq 0$
- interpolates only at $\mathbf{p}_0$ and $\mathbf{p}_3$
  - $b_0 = 1$ at $u = 0$, $b_3 = 1$ at $u = 1$
  - b_1 and b_2 never reach 1
- the basis functions are everywhere non-zero, except at the end points $\Rightarrow$ the control points do not exert local control
- the curves are symmetric: reversing the control points yields the same curve



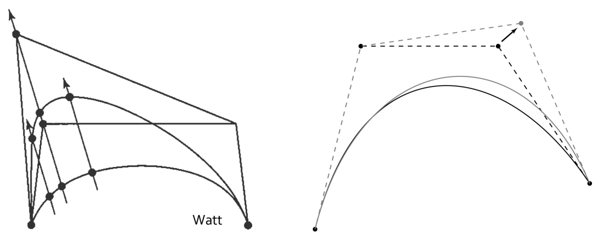


Durand,Hodgins

Non-Local Control

Every control point affects every point on the curve (except the endpoints)

Moving a single control point affects the whole curve!

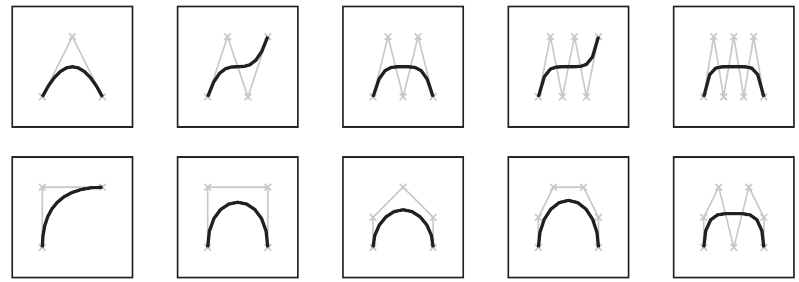


Curless

Variation Diminishing Property

Bézier curves have the variation diminishing property: each is no more "wiggly" than its control polygon
⇒ does not cross a line more than its control polygon

Various Bézier curves, of degrees 2-6:



Shirley

Bernstein Basis Polynomials

The blending/basis functions for Bézier curves can in general be expressed as the Bernstein basis polynomials:

$$b_{k,n}(u) = \binom{n}{k} u^k (1-u)^{n-k} = \frac{n!}{k!(n-k)!} u^k (1-u)^{n-k}$$

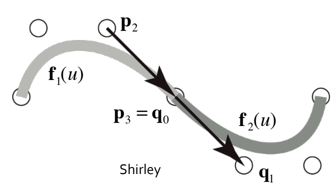
Bézier curve eqn: $\mathbf{f}(u) = \sum_{k=0}^{n} \frac{n!}{k!(n-k)!} u^k (1-u)^{n-k} \mathbf{p}_k$

Shirley

Joining Bézier Curves

Multiple-segment cubic Bézier curve can achieve

- G^1 continuity if: $\mathbf{q}_0 = \mathbf{f}_2(0) = \mathbf{f}_1(1) = \mathbf{p}_3$ and $(\mathbf{q}_1 - \mathbf{q}_0) = k(\mathbf{p}_3 - \mathbf{p}_2)$, the three points ($\mathbf{p}_2$, $\mathbf{p}_3 = \mathbf{q}_0$, and $\mathbf{q}_1$) are collinear
  - if you changed one of these three, you must change the others, but only need to change these three, not $\mathbf{p}_1$ for example ⇒ local support



- C^1 continuity if $k = 1$
- can't guarantee C^2 or higher continuity
  - each additional degree of continuity restricts the position of an additional control point → cubic Bézier has none to spare

Bézier Curve/Surface Problems

To make a long continuous curve with Bézier segments requires using many segments

Maintaining continuity requires constraints on the control point positions

- the user cannot arbitrarily move control points and automatically maintain continuity
- the constraints must be explicitly maintained
- it is not intuitive to have control points that are not free

Consider: B-spline