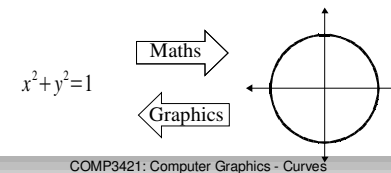


## Curves

- We want to be able to represent and display curves as well as straight lines.
- Require a representation that is
  - Powerful – can create all the curve shapes we want
  - Efficient – uses as little computing resources as possible

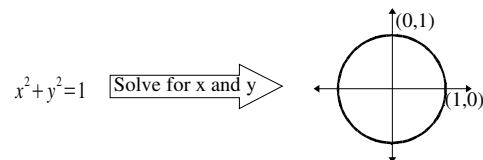
## Curves in Maths

- When you learned about curve drawing in maths, you started with an equation and drew the curve to understand the equation
- In graphics we want to find an equation to produce the curve shape we want.



## Implicit Representation

- An implicit representation of a curve is an equation that must be solved to find all the points on the curve.
- Not convenient



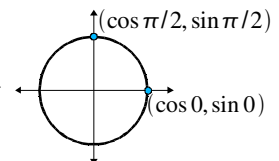
## Parametric Representation

- Curve is defined by a function of a parameter  $t$

$$C(t) = (x(t), y(t)), a \leq t \leq b$$

Example: parametric equation of a circle

$$C(t) = (\cos t, \sin t), 0 \leq t \leq 2\pi$$



## Drawing a parametric curve

```
gl.glBegin( GL.GL_LINE_STRIP);  
for (int i = 0; i<= 20; i++){  
    double angle = (i/20.0)*2*Math.PI;  
    gl.glVertex2d(Math.cos(angle),  
                 Math.sin(angle));  
}  
gl.glEnd();
```

## What sort of function to use?

- Linear  $a+bt$  just get a straight line
- Quadratic  $a+bt+ct^2$  doesn't have enough flexibility
- Cubic  $a+bt+ct^2+dt^3$  works well
- Quartic  $a+bt+ct^2+dt^3+et^4$  unnecessarily complicated

## Control Points

- Don't want to have to write out equations, so define curves in terms of control points
- Curve will interpolate (go through) or approximate (go near) control points

## Natural Cubic Splines

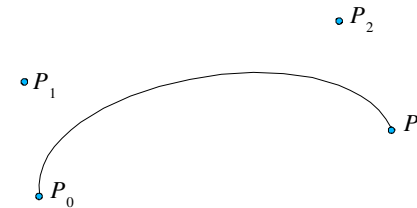
- Interpolate all control points
- A cubic curve between each pair of control points ("piecewise cubic")
- Four unknowns in a cubic function
  - Going through two control points gives two equations
  - requiring that derivatives match at end points of curves gives two more equations

## Natural Cubic Splines

- But ...
  - While curve goes through control points, middle parts can bulge out a lot
  - Moving one control point causes entire curve to change – no local control over shape of curve
- Can do better if we allow curve to approximate control points

## Bezier Curve

- Defined by four control points
- $B(t) = b_0(t)P_0 + b_1(t)P_1 + b_2(t)P_2 + b_3(t)P_3, 0 \leq t \leq 1$



## Bernstein Polynomials

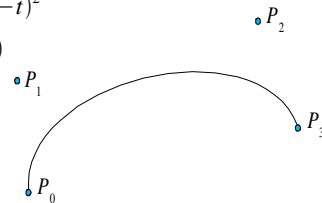
$$B(t) = b_0(t)P_0 + b_1(t)P_1 + b_2(t)P_2 + b_3(t)P_3, 0 \leq t \leq 1$$

$$b_0(t) = 1 - 3t + 3t^2 - t^3 = (1-t)^3$$

$$b_1(t) = 3t - 6t^2 + 3t^3 = 3t(1-t)^2$$

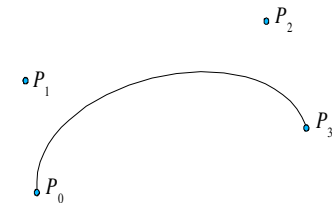
$$b_2(t) = 3t^2 - 3t^3 = 3t^2(1-t)$$

$$b_3(t) = t^3$$



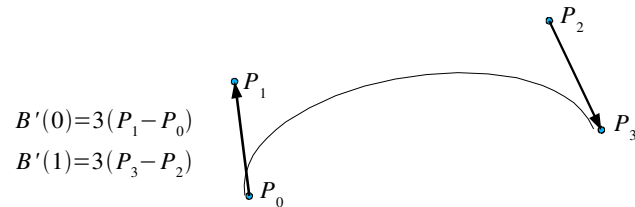
## Bezier Properties

- Interpolates  $P_0$  and  $P_3$
- Approximates  $P_1$  and  $P_2$
- OHP



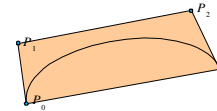
## Tangent Property

- Tangent at  $P_0$  is line from  $P_0$  to  $P_1$
- Tangent at  $P_3$  is line from  $P_2$  to  $P_3$
- OHP



## Convex hull property

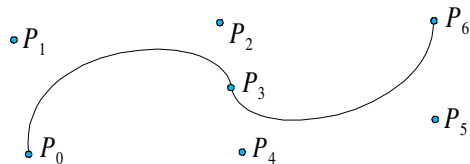
- $b_0(t) + b_1(t) + b_2(t) + b_3(t) = 1$
- $b_i(t) \geq 0$



- Points on curve are a *convex combination* of control points, so curve must lie inside *convex hull* of control points
- So curve does not bulge out too much

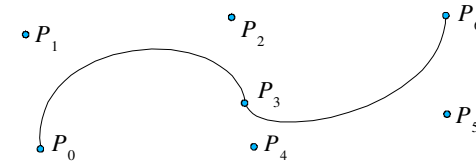
## Joining without corners

- If the first derivative of a curve is continuous, we say it has  $C^1$  continuity.
- Can join  $B(P_0, P_1, P_2, P_3)$  and  $B(P_3, P_4, P_5, P_6)$  with  $C^1$  continuity if  $P_3 - P_2 = P_4 - P_3$



## Geometric Continuity

- If the magnitude of the derivative changes but the direction doesn't, there still won't be a corner at the junction
- This is called  $G^1$  continuity
- Just need  $P_2, P_3, P_4$  to be collinear



## De Casteljau algorithm

$$P_{01} = (P_0 + P_1)/2$$

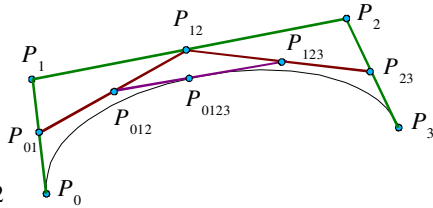
$$P_{12} = (P_1 + P_2)/2$$

$$P_{23} = (P_2 + P_3)/2$$

$$P_{012} = (P_{01} + P_{12})/2$$

$$P_{123} = (P_{12} + P_{23})/2$$

$$P_{0123} = (P_{012} + P_{123})/2$$



$$B(P_0, P_1, P_2, P_3) = B(P_0, P_{01}, P_{012}, P_{0123}) \cup B(P_{0123}, P_{123}, P_{23}, P_3)$$

## Osculating Circle

- The *osculating circle* to a curve at a point  $P$  is the circle that just touches it at  $P$
- Three points define a circle (why?). The osculating circle is the limit of the circles through  $P$ ,  $Q$  and  $R$  where  $Q$  and  $R$  are points on the curve that approach  $P$ .



## Curvature

- The curvature of a circle with radius  $r$  is  $1/r$
- Small circles bend more sharply and have a higher curvature
- The curvature of a curve at a point  $P$  is the curvature of the osculating circle
- Imagine driving a car along the curve – the curvature corresponds to steering wheel position

## Second derivative

- The tangent to a curve at a point has the same first derivative as the curve
- The osculating circle has the same first and second derivative as the curve

## $C^2$ and $G^2$ Continuity

- A curve is  $C^2$  continuous if it is  $C^1$  continuous and the second derivative is continuous.
- A curve is  $G^2$  continuous if it is  $G^1$  continuous and the curvature is continuous.
- It is not physically possible to instantly change the steering wheel of a car from one position to another, so curves must be  $G^2$  if you want to be able to drive along them.

## Uniform B Splines

- To get curves that join with  $C^2$  continuity you must give up on interpolating any control points.
- Uniform B splines approximate all the control points

## Uniform B Spline definition

$$B_0(t) = b_0(t)P_0 + b_1(t)P_1 + b_2(t)P_2 + b_3(t)P_3$$

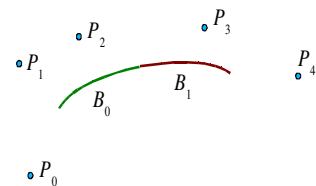
$$B_1(t) = b_0(t)P_1 + b_1(t)P_2 + b_2(t)P_3 + b_3(t)P_4$$

$$b_0(t) = (1 - 3t + 3t^2 - t^3)/6$$

$$b_1(t) = (4 - 6t^2 + 3t^3)/6$$

$$b_2(t) = (1 + 3t + 3t^2 - 3t^3)/6$$

$$b_3(t) = t^3/6$$



## Uniform B Splines

- Uniform B Splines join with  $C^2$  continuity
- Proof: OHP

## Non-uniform B splines

- Can define B spline as one function on range

$$0 \leq t \leq 3$$
$$B(t) = \begin{cases} B_0(t), & 0 \leq t \leq 1 \\ B_1(t-1), & 1 < t \leq 2 \\ B_2(t-2), & 2 < t \leq 3 \end{cases}$$

- Values 0,1,2,3 where the function changes are called *knots*. These ones are uniformly spaced, but can generalise to non-uniform spacing.
- Knots 0,0,0,0,1,1,1,1 produce the Bezier curve.

## Rational B splines

- To be able to produce a circle need to be able to weight each point. Use  $w$  component in homogenous coordinates.  $\begin{bmatrix} x \\ y \\ w \end{bmatrix}$
- Resulting function  $(x(t)/w(t), y(t)/w(t))$  is a rational polynomial (ratio of two cubics).
- Larger weights pull curve in that direction

## NURBS

- Non-Uniform Rational B Splines
- Generalise all the curves we have seen so far
- Don't need special cases for circles, Beziers or Uniform B Splines – they are all just special NURBS